CENTRO BRASILEIRO DE PESQUISAS FÍSICAS – CBPF

DESENVOLVIMENTO DE UM DISPOSITIVO INJETOR DE CARGA PARA VALIDAÇÃO DA ELETRÔNICA DE *FRONT-END* DO DETECTOR DE FIBRAS CINTILANTES DO LHCB

Ulisses de Freitas Carneiro da Graça

Orientador:
Prof. Dr. André Massafferri Rodrigues
CBPF

Co-orientador:
Prof. Dr. Antonio Pellegrino
University of Groningen

Rio de Janeiro
Fevereiro de 2018

CENTRO BRASILEIRO DE PESQUISAS FÍSICAS – CBPF

DESENVOLVIMENTO DE UM DISPOSITIVO INJETOR DE CARGA PARA VALIDAÇÃO DA ELETRÔNICA DE *FRONT-END* DO DETECTOR DE FIBRAS CINTILANTES DO LHCB

Ulisses de Freitas Carneiro da Graça

Dissertação de mestrado apresentada ao
Centro Brasileiro de Pesquisas Físicas
como parte dos requisitos necessários à
obtenção do título de Mestre em Física

Aprovado por banca examinadora mediante defesa oral
pública realizada às 15 horas do dia 09 de março de 2018, no
auditório do 2º andar do CBPF, Rua Xavier Sigaud 150, Botafogo

Orientador:
Prof. Dr. André Massafferri Rodrigues
CBPF

Co-orientador:
Prof. Dr. Antonio Pellegrino
University of Groningen

Rio de Janeiro
Fevereiro de 2018

G729 Graça, Ulisses de Freitas Carneiro da
Desenvolvimento de um Dispositivo Injetor de Carga para validação da eletrônica de front-end do detector de fibras cintilantes do LHCb / — 2018.
xii, 126f. + apêndices e anexo : il.color. , grafs. , tabs. ; enc.

Dissertação (Mestrado) , 2018.
Bibliografia : f.121-126
Orientadores : André Massafferri Rodrigues
Antonio Pellegrino

1. Instrumentação Científica. 2. Engenharia Eletrônica. 3. Física de Altas Energias. 4. Detectores de Partículas. 5. Aquisição de Dados. I. André Massafferri Rodrigues (Orientador). II. Antonio Pellegrino (Coorientador). III. Título.

CDD 621.38

DEDICATÓRIA

Dedico esta obra a todos os que fazem parte da minha vida e que eu tanto amo, em especial meu pai, Luiz Cláudio Carneiro da Graça, minha mãe, Dirce Maria de Freitas e minha namorada, Camila Almeida da Silva.

AGRADECIMENTOS

Agradeço a todos os que de certa forma colaboraram com este trabalho, em especial a meus colegas de trabalho do IEN e do CEFET/RJ, pelo apoio e compreensão, a meus orientadores André Massafferri e Antonio Pellegrino, pelos inestimáveis ensinamentos, e aos colaboradores dos institutos parceiros, pelo irrestrito suporte prestado com grande competência.

RESUMO

O complexo de aceleradores do *CERN* está sendo atualizado para aumentar a luminosidade de feixe do *LHC* de modo que seus experimentos possam adquirir dados a uma taxa ainda maior, proporcionando avanços significativos na pesquisa científica. O experimento *LHCb* será atualizado para que possa operar sem perda de eficiência na nova luminosidade. Para tanto, todo seu sistema de trajetografia precisou ser reprojetado com base numa nova tecnologia de detecção, com aumento substancial do número de canais. Novos dispositivos eletrônicos para a aquisição de dados precisaram ser desenvolvidos e estão agora sendo produzidos em grandes quantidades. Um sistema automático de teste deve ser construído para viabilizar o comissionamento e a caracterização em grande escala destes novos dispositivos. A presente dissertação descreve o desenvolvimento de um Dispositivo Injetor de Carga, parte do sistema de teste da eletrônica de *front-end* das estações de fibras ópticas cintilantes que integrarão o novo sistema de trajetografia do *LHCb*.

Palavras-chave: Instrumentação Científica; Engenharia Eletrônica; Física de Altas Energias; Detectores de Partículas; Aquisição de Dados.

ABSTRACT

CERN's accelerator complex is being updated in order to increase the beam luminosity of the LHC so that its experiments can acquire data at an even higher rate, providing significant advances in scientific research. The LHCb experiment will be updated to operate without loss of efficiency with the new luminosity. In order to do so, its entire Tracker System had to be rebuilt using a new detection technology, with a substantial increase in the number of channels. New electronic devices for data acquisition were developed and are now being produced in large quantities. An automatic testing system must be built to enable large-scale commissioning and characterization of these new devices. This thesis describes the development of a Charge Injector Device, part of the front-end electronics testing system of the scintillating optical fiber stations that will integrate the new LHCb Tracker System.

Keywords: Scientific Instrumentation; Electronic Engineering; High Energy Physics; Particle Detectors; Data acquisition.

SUMÁRIO

Introdução **1**

Capítulo 1
O *CERN* e o Experimento *LHCb* **5**
1.1. O Complexo de Aceleradores do ***CERN*** 5
1.2. O Experimento ***LHCb*** 7
1.2.1. Reconstrução de Trajetória no ***LHCb*** 11
1.3. Motivação para a Atualização do ***LHCb*** 12

Capítulo 2
SciFi: Um Novo Detector Para o *LHCb* **15**
2.1. As Atuais Estações ***T*** 15
2.2. Motivação para o Projeto ***SciFi*** 16
2.3. Principais Desafios do Projeto ***SciFi*** 17
2.3.1. Performance 17
2.3.2. Geometria 18
2.3.3. Operação em Ambiente Radioativo 19
2.3.4. Resfriamento 20
2.4. As Novas Estações ***SciFi*** 20
2.5. Fibras Ópticas Cintilantes 23
2.6. Fotomultiplicadoras de Silício (***SiPM's***) 27
2.7. Mecanismo de Leitura 30
2.8. Caixas de Leitura (***Read-out Boxes***) 32
2.8.1. Eletrônica ***Front-End*** do ***SciFi*** 34
2.8.2. ASIC **PACIFIC** 37
2.8.3. Módulo de Clusterização 40
2.8.4. Master Board 42
2.9. MiniDAQ 43

Capítulo 3
Sistema de Teste da Eletrônica de *Front-End* **45**
3.1. Objetivos do Sistema de Teste 45
3.2. Conceituação do Sistema de Teste 46
3.3. Arquitetura do Sistema de Teste 48
3.4. Dispositivo Injetor de Carga 49

3.5. O Módulo Injetor 52
3.5.1. Circuitos Digitais 53
3.5.2. Circuitos Analógicos 56
3.5.3. Cabeamento das Saídas de Pulso 60
3.6. Desenvolvimento do Protótipo de 8 Canais 61
3.7. Desenvolvimento do Módulo Injetor de 256 Canais 64
3.7.1. Dimensionamento das Trilhas 69
3.8. Avaliação do Módulo Injetor de 256 Canais 76
3.9. Controle Lento do Módulo Injetor 77
3.9.1. Adaptador ***I2C - USB*** 78
3.9.2. Controle pelo Sistema SCADA 79
3.10. Controle Rápido do Módulo Injetor 81
3.11. Módulo de Controle 82
3.12. Montagem Mecânica 83

Capítulo 4
Resultados Obtidos 85
4.1. Primeira Sequência de Testes 85
4.1.1. Ferramentas e Metodologia 85
4.1.2. Análise da Saída do Injetor 87
4.1.3. Análise da Resposta do ***ASIC*** **PACIFIC** 90
4.2. Segunda Sequência de Testes 96
4.2.1. Ferramentas e Metodologia 96
4.2.2. Automação da Aquisição de Dados 98
4.2.3. Processamento dos Dados 103
4.2.4. Avaliação das Saídas do Módulo Injetor 105

Capítulo 5
Conclusões 117
5.1. Produção do Lote Piloto do Módulo Injetor 117
5.2. Interpretação dos Resultados Obtidos 118
5.3. Continuidade do Sistema de Teste 119
5.4. Comissionamento do SciFi 119
5.5. Produção Acadêmica Relacionada 119

Referências Bibliográficas 121

Anexo A
Publicação TWEPP 2016 127

Apêndice A
Guia de Início Rápido do Módulo Injetor 139
Parte I – Operação em ***Standalone Mode*** 140
Parte II – Operação em ***MiniDAQ Mode*** 143

Apêndice B
Diagramas Esquemáticos do Injetor de Carga 147

Apêndice C
Desenhos da PCB do Módulo Injetor 169

Apêndice D
FIRMWARE do FPGA do Injetor de Carga 185

Apêndice E
Código C^{++} da Seção 3.9.1 219

Apêndice F
Formas de Onda na Saída do Injetor 227

Apêndice G
Dados Tabulados da Seção 4.1.2 263

Apêndice H
Código C^{++} da Seção 4.2.2 267

Apêndice I
***Script Python* para Formatação dos Dados 283**

Apêndice J
***Script Python* para Análise dos Dados 287**

Apêndice K
Histogramas da Seção 4.2.4 313

LISTA DE FIGURAS

Figura 1 - Complexo de aceleradores do CERN. 5

Figura 2 - Linhas de feixe do *LHC*. 6

Figura 3 - Localização do experimento *LHCb*. 7

Figura 4 - Distribuição angular da produção de mésons B 8

Figura 5 - Experimento *LHCb* atual 8

Figura 6 - Mecanismo para medida do *momentum* de partículas carregadas. 10

Figura 7 - Componentes das atuais estações *T*. 10

Figura 8 - Tipos de trajetórias 11

Figura 9 - Eficiência do *trigger* L0 em função da luminosidade 12

Figura 10 - Expectativa de dose acumulada.....19

Figura 11 - As três estações de fibras cintilantes 21

Figura 12 - Representação em vistas de uma estação SciFi. 22

Figura 13 – Dimensões de um Módulo e definição dos ângulos *stereo*.....22

Figura 14 – Estrutura de um plano de detecção composto por 12 módulos 22

Figura 15 - Representação da estrutura de uma fibra cintilante 23

Figura 16 - Plano de fibras cintilantes demonstrado por *B. Beischer et al*.....25

Figura 17 – Vista em corte transversal dos planos de detecção.....26

Figura 18 - Protótipo do plano detector das estações *SciFi*.....26

Figura 19 - Detalhes da construção da fotomultiplicadora de silício 27

Figura 20 - Típico sinal de saída de uma fotomultiplicadora de silício.....28

Figura 21 - Acoplamento do plano de fibras cintilantes na *SiPM*. 30

Figura 22 - Leitura do ponto de passagem de uma partícula 31

Figura 23 - Detalhe das conexões entre a caixa de leitura e o plano de fibras 31

Figura 24 - *Design* da caixa de leitura 32

Figura 25 - Aparência final das caixas de leitura 33

Figura 26 - Diagrama do fluxo de dados na eletrônica *front-end*..... 34
Figura 27 - Diagrama em blocos de uma caixa de leitura..... 35
Figura 28 - Montagem (parcial) do protótipo da eletrônica *front-end*..... 36
Figura 29 – Esquema da eletrônica de 1/2 caixa de leitura..... 36
Figura 30 – Diagrama em blocos de um canal analógico no **PACIFIC**..... 37
Figura 31 - Diagrama esquemático do estágio de entrada..... 38
Figura 32 - Diagrama esquemático do estágio de filtragem..... 39
Figura 33 - Diagrama esquemático do estágio de integração..... 39
Figura 34 - Representação gráfica do algoritmo de clusterização..... 41
Figura 35 – Fotografia do *MiniDAQ*..... 43
Figura 36 – Diagrama do conceito "*SiPM* multiplexada"..... 46
Figura 37 – Diagrama do conceito "Gerador com *fan-out* comutado por relé"..... 47
Figura 38 – Diagrama do conceito "Dispositivo Injetor de Carga"..... 47
Figura 39 – Diagrama em blocos do sistema de testes..... 48
Figura 40 – Dispositivo Injetor de Carga..... 50
Figura 41 – Montagem parcial do Dispositivo Injetor de Carga..... 51
Figura 42 – Diagrama em blocos do Módulo Injetor..... 52
Figura 43 – Esquema de controle dos *DAC's*..... 55
Figura 44 – Conversores digital-analógico..... 56
Figura 45 – Modulador de Amplitude..... 57
Figura 46 – *Buffer* de Saída..... 58
Figura 47 – Reguladores de tensão $6v$ e $3v3$..... 59
Figura 48 – Reguladores de tensão $1v2$ e $2v5$..... 59
Figura 49 – Esquema de cabeamento do Dispositivo Injetor de Carga..... 60
Figura 50 – Protótipo com 8 Canais, desenvolvido como uma prova de conceito..... 61
Figura 51 – Bancada de avaliação do protótipo..... 62
Figura 52 – Revisões após a avaliação do protótipo..... 63
Figura 53 – Exemplar do Injetor de 256 Canais..... 64

Figura 54 – Detalhe da organização em grupos do Módulo Injetor. 65
Figura 55 – Transferência de *design* a partir do protótipo. 65
Figura 56 – Vista Superior do Injetor de 256 Canais. 66
Figura 57 – Face Inferior do Módulo Injetor. 67
Figura 58 – Detalhe do FPGA, o concentrador do Processamento Digital. 67
Figura 59 – Detalhe dos Circuitos Reguladores de Tensão. 68
Figura 60 – Montagem dos grupos analógicos. 68
Figura 61 – Roteamento dos sinais de disparo dos canais. 70
Figura 62 – Roteamento das saídas do Módulo Injetor. 70
Figura 63 – Roteamento das linhas seriais de seleção dos *DAC's.* 71
Figura 64 – Roteamento do *clock* serial para seleção dos *DAC's.* 71
Figura 65 – Roteamento do *latch clock* para seleção dos *DAC's.* 72
Figura 66 – Roteamento de reset dos decodificadores de seleção dos *DAC's.* 72
Figura 67 – Roteamento dos sinais de seleção dos *DAC's* 72
Figura 68 – Resumo do roteamento para seleção dos *DAC's.* 73
Figura 69 – Roteamento das linhas seriais de controle dos *DAC's.* 73
Figura 70 – Roteamento das linhas de *clock* dos *DAC's.* 74
Figura 71 – Roteamento de *clock* distribuído para os *DAC's* 74
Figura 72 – Roteamento das linhas de *latch* dos *DAC's.* 74
Figura 73 – Roteamento de *latch* distribuído para os *DAC's.* 75
Figura 74 – Resumo do roteamento de controle dos *DAC's.* 75
Figura 75 – Adaptador I^2C/USB desenvolvido para controle do Módulo Injetor 78
Figura 76 – Interface de usuário do adaptador I^2C/USB no terminal serial. 78
Figura 77 – Diagrama em blocos do caminho de controle pelo sistema SCADA. 79
Figura 78 – Interface de usuário da aplicação SCADA programada no WinCC. 80
Figura 79 – Diagrama em blocos do mecanismo de controle rápido. 81
Figura 80 – Módulo de Controle do Dispositivo Injetor de Carga. 82
Figura 81 – Vista tridimensional do projeto da montagem mecânica. 83

Figura 82 – Vista frontal da montagem mecânica. 83
Figura 83 – Bancada de testes. 86
Figura 84 – Pontas de prova conectadas às portas de *DEBUG* do **PACIFIC**. 86
Figura 85 – Detalhe da conexão das pontas de prova ativas. 87
Figura 86 – Área do pulso em função de *aDAC*, para diversos valores de *wDAC*. 88
Figura 87 – Carga de saída em função de *aDAC*, para diversos valores de *wDAC*.... 88
Figura 88 –Tensão de pico do pulso em função de *aDAC*, para vários *wDAC*..... 89
Figura 89 – Largura do pulso em função de *aDAC*, para vários *wDAC*. 89
Figura 90 – Tela de osciloscópio com os sinais *PACIFIC DEBUG* 91
Figura 91 – Amplitude do *DEBUG_ TH* em função da amplitude do pulso 92
Figura 92 – Comportamento do sinal *DEBUG_ TH* em função da amplitude 92
Figura 93 – Amplitude de *DEBUG_ TH* em função da carga injetada 93
Figura 94 – Detalhe da rampa presente no sinal *DEBUG_ TH*. 94
Figura 95 – Sinal *DEBUG_ PRE* em função da amplitude do pulso 95
Figura 96 – Amplitude do *DEBUG_ PRE* em função da amplitude do pulso..... 95
Figura 97 – Bancada da segunda sequência de testes. 97
Figura 98 – Detalhe da conexão do Módulo Injetor no osciloscópio. 97
Figura 99 – Algoritmo de teste do Módulo Injetor. 99
Figura 100 – Diagrama em blocos da implementação do algoritmo de teste..... 100
Figura 101 – *Hardware* utilizado para a implementação do algoritmo de teste. 100
Figura 102 – Captura de tela da interface de usuário do algoritmo de teste. 101
Figura 103 – Diretório com os arquivos produzidos pelo osciloscópio. 101
Figura 104 – Exemplo de arquivo salvo em disco com a captura de tela. 102
Figura 105 – Exemplo de arquivo salvo em disco com o resultado das medições..... 102
Figura 106 – Fluxograma de tratamento e análise dos dados. 103
Figura 107 – Detalhe do *baseline* deslocado do sinal 104
Figura 108 – Área corrigida do pulso em função de *aDAC* no canal 0x05. 106
Figura 109 – Carga calculada do pulso em função de *aDAC* no canal 0x05. 106

Figura 110 – Largura do pulso em função de $aDAC$ no canal 0x05. 107
Figura 111 – Atraso de disparo do pulso em função de $aDAC$ no canal 0x06. 107
Figura 112 – Tensão de pico do pulso em função de $aDAC$ no canal 0x05. 108
Figura 113 – Distribuição dos parâmetros de ajuste 109
Figura 114 – Área corrigida do pulso em função de $wDAC$ no canal 0x05. 111
Figura 115 – Carga calculada do pulso em função de wDAC no canal 0x05 111
Figura 116 – Largura do pulso em função de $wDAC$ no canal 0x05. 112
Figura 117 – Atraso de disparo do pulso em função de $wDAC$ no canal 0x06. 112
Figura 118 – Tensão de pico do pulso em função de $wDAC$ no canal 0x05. 113
Figura 119 – Distribuição dos parâmetros de ajuste 113
Figura 120 – Distribuições de atrasos intercanal para diversos valores de $aDAC$. ... 115
Figura 121 – Conjuntos de dados dos histogramas destacados. 116
Figura 122 – Conjuntos de dados dos histogramas destacados. 116

LISTA DE TABELAS

Tabela 1 – Carga do pulso em função de diferentes parâmetros 90

Tabela 2 – Parâmetros de pulso em função de $aDAC$ 110

Tabela 3 – Carga do pulso em função de diferentes parâmetros 114

Introdução

A física de partículas é uma área da ciência que tem como principais objetivos compreender a estrutura da matéria e as interações decorrentes das forças fundamentais. A teoria que até agora descreve com maior sucesso as interações entre partículas subatômicas é o Modelo Padrão (*Standard Model - SM*) [1]. A validade do *SM* é confirmada pela verificação experimental de partículas que foram previstas por esta teoria, como exemplo, o bóson de *Higgs* [2]. Até agora, as previsões do *SM* estão de acordo com diversas observações experimentais realizadas. Entretanto, ainda permanecem algumas questões fundamentais sem resposta, notoriamente o mistério da antimatéria desaparecida [3] e a natureza da matéria escura [4]. Algumas teorias que vão além do *SM* tentam resolver estas questões.

Diversos experimentos são realizados com o objetivo de confirmar previsões teóricas e procurar fenômenos ainda não explicados através do exame de novas teorias. Em ambiente controlado, isto é possível graças à colisão de partículas em altas energias. No *Large Hadron Collider* (*LHC*), instalado no *CERN* próximo a Genebra, na Suíça, prótons são acelerados e colidem com energia combinada de centro de massa de até $13\,TeV$. O *LHC* hospeda quatro grandes experimentos, otimizados para diferentes estudos. O experimento otimizado para o estudo de Mésons-B é o *LHCb*.

O experimento *LHCb* registrou mais de $3\,fb^{-1}$ de dados durante os primeiros anos de execução (*Run 1*). As partículas estudadas pelo experimento originam-se de um ponto de interação conhecido como vértice primário, no qual ocorrem sucessivas colisões entre prótons acelerados pelo *LHC*. A detecção do percurso das partículas produzidas nestas colisões é ferramenta essencial para o experimento, pois permite reconstruir suas trajetórias. É possível ainda observar a deflexão causada pelo campo magnético do *LHCb* e obter uma medida precisa do *momentum* das partículas carregadas que

por ele passaram. Para maximizar sua sensibilidade, o experimento deve ser realizado com a maior energia e a maior luminosidade de feixe possíveis do acelerador. Um conjunto detector com capacidade de rastreamento superior também é necessário para que se possa medir com a maior resolução possível os fenômenos estudados.

Um incremento na taxa de aquisição de dados proporcionará avanços significativos para a pesquisa científica no *LHCb*. Isto requer, entretanto, a completa atualização do experimento. Um upgrade foi então proposto para o próximo desligamento de longa duração (*Long Shutdown 2* - LS2) do *LHC*, entre meados de 2018 e o final de 2019. Durante os cinco anos que se sucederão ao religamento ocorrerá o terceiro período de execução (*Run 3*), com uma luminosidade 5 vezes maior do feixe. O incremento da luminosidade, combinado com um desempenho melhorado do *LHCb*, deverá ser capaz de aumentar em 10 vezes a quantidade de dados adquiridos, o que melhorará significativamente a estatística associada aos fenômenos físicos estudados.

Todos os detectores do *LHCb* serão atualizados para lidar com a nova luminosidade disponível, tornando-o capaz de capturar dados de todas as colisões. As atuais estações de *downstream tracker* do experimento (*T-stations*) serão substituídas por estações do tipo *Scintillating Fiber Tracker* (*SFT* ou, simplesmente, *SciFi*), cuja tecnologia de detecção é baseada em fibras cintilantes de $250\mu m$ de diâmetro lidas por fotomultiplicadoras de silício (*SiPM's*), posicionadas fora da área sensível do detector.

Um importante elemento do projeto *SciFi* são as caixas de leitura, ou *Readout Boxes* (*ROB´s*), que estão sendo produzidos em massa para instalação junto ao detector. Sofisticados circuitos eletrônicos de *front-end* contidos nestas caixas adquirem e digitalizam os sinais elétricos provenientes das *SiPM's*, enviando-os no formato adequado através do restante dos sistemas de *data acquisition* (*DAQ*). Desta forma, dados

adquiridos no detector podem ser processados no *back-end*[1] do experimento, longe da radiação ionizante produzida pelo *LHC*.

Esta dissertação trata do desenvolvimento de um Dispositivo Injetor de Carga de 2.048 canais, elemento crucial do sistema de controle de qualidade dos circuitos eletrônicos de *front-end*, responsabilidade do CBPF dentro do projeto *SciFi*. A garantia do controle de qualidade das ≈ 300 caixas de leitura em produção requer a validação e a caracterização da eletrônica de *front-end* de cada uma delas.

No Capítulo 1 o complexo de aceleradores do *CERN* e o experimento *LHCb* serão descritos. Os principais subsistemas do experimento *LHCb* e os motivos para sua atualização serão apresentados.

As principais características do projeto *SciFi* serão discutidas no Capítulo 2, juntamente com estudos sobre as principais características que foram levadas em conta para a definição do sistema de teste proposto.

O sistema de teste - tema desta dissertação - será apresentado no Capítulo 3, onde também serão detalhadamente descritos cada um de seus elementos.

Uma série de testes foi conduzida para certificar-se de que o Dispositivo Injetor é capaz de excitar corretamente a eletrônica de *front-end*. No Capítulo 4 a metodologia e os resultados destes testes serão apresentados e discutidos.

Finalmente, considerações finais e recomendações para trabalhos futuros serão apresentadas no Capítulo 5.

---

[1] O termo *back-end* se refere ao conjunto de elementos do sistema de *data acquisition* que não se encontra localizado na vizinhança imediata dos detectores.

Capítulo 1
O *CERN* e o Experimento *LHCb*

1.1. O Complexo de Aceleradores do *CERN*

A Organização Europeia de Pesquisa Nuclear, conhecida como *CERN*, é atualmente o maior centro para pesquisa em física de partículas do mundo. O complexo de aceleradores do *CERN* está representado na Figura 1.

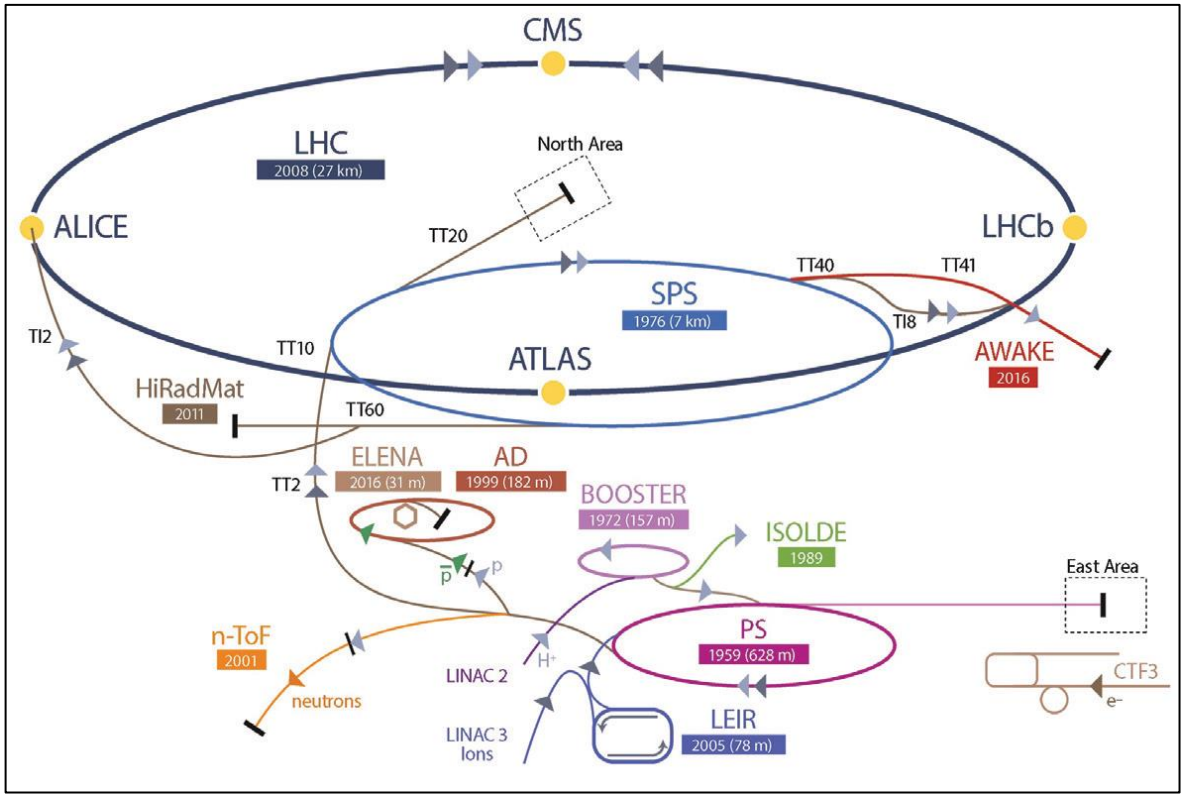


Figura 1 - Complexo de aceleradores do CERN.

Este complexo de aceleradores abriga desde experimentos a energias relativamente baixas até experimentos que utilizam a mais alta energia, disponível no *Large Hadron Collider* (*LHC*).

Os prótons acelerados pelo *LHC* provêm de uma complexa rede de aceleradores. O processo se inicia no acelerador linear *LINAC2*, onde prótons extraídos de uma fonte

de gás hidrogênio são acelerados produzindo um feixe com energia de $50\,MeV$. Este feixe é acelerado para $1{,}4\,GeV$ através do *Proton Synchrotron Booster* e, em seguida, é transferido ao *Proton Synchrotron* (*PS*) onde atinge a energia de $26\,GeV$. Na sequência, o *Super Proton Synchrotron* (*SPS*) acelera e armazena o feixe com energia de $450\,GeV$. A partir deste ponto, direcionadores tangenciais podem desviar o feixe proveniente do *SPS* para áreas *test-beam* localizadas no Setor Norte (no centro do *LHC*) ou injetá-lo na forma de pacotes em um par de feixes com sentidos opostos no *LHC*, onde atualmente partículas aceleradas podem atingir energia combinada de centro de massa de até $\sqrt{s} = 13\,TeV$, com luminosidade instantânea $\mathcal{L}_{máxima} \approx 10^{34}\,cm^{-2}s^{-1}$. Os dois feixes ocupam linhas separadas do *LHC* (Figura 2) e são postos a colidir com intervalos regulares de $25\,ns$ em cruzamentos de feixe conhecidos como *bunch-crossings*. Quatro grandes experimentos ocorrem atualmente no *LHC*, um deles é o *LHCb* (Figura 3).

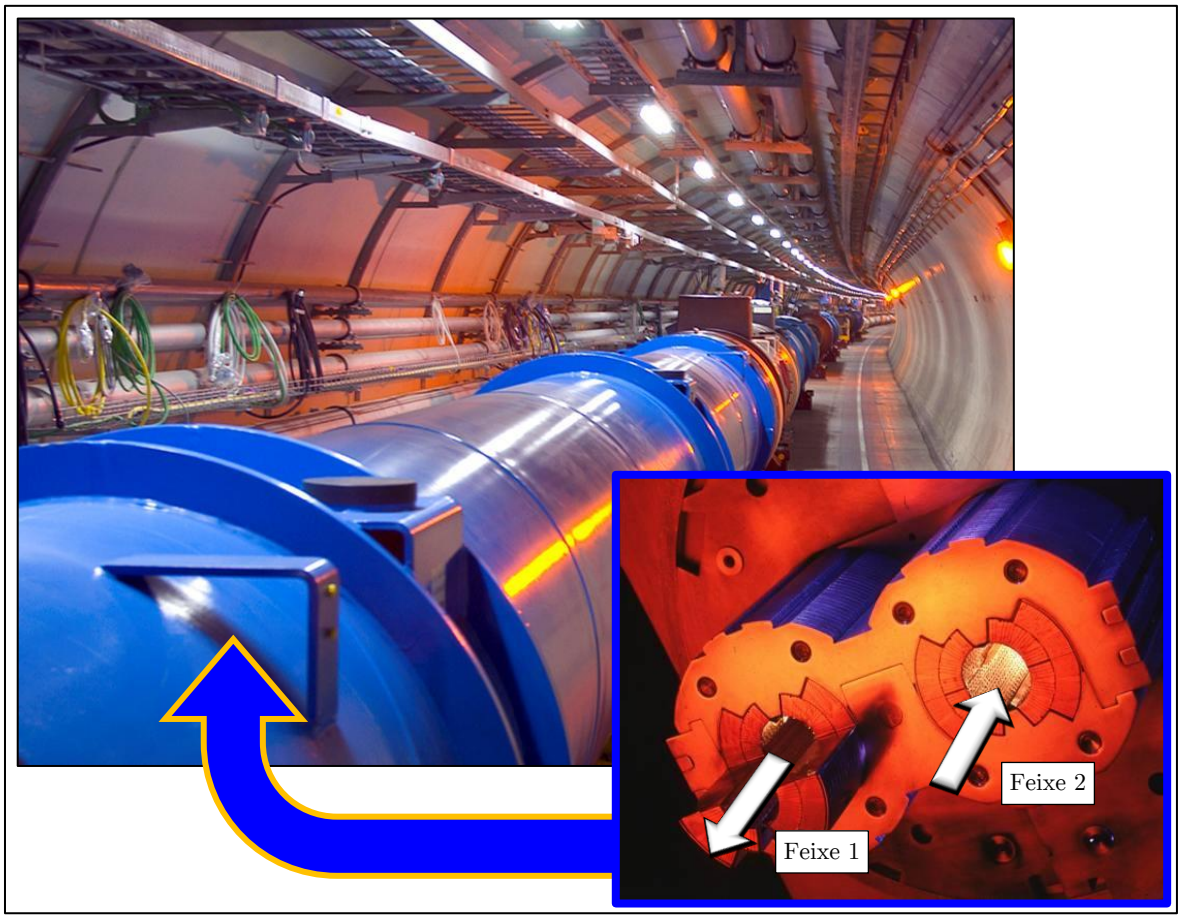


Figura 2 - Linhas de feixe do *LHC*.

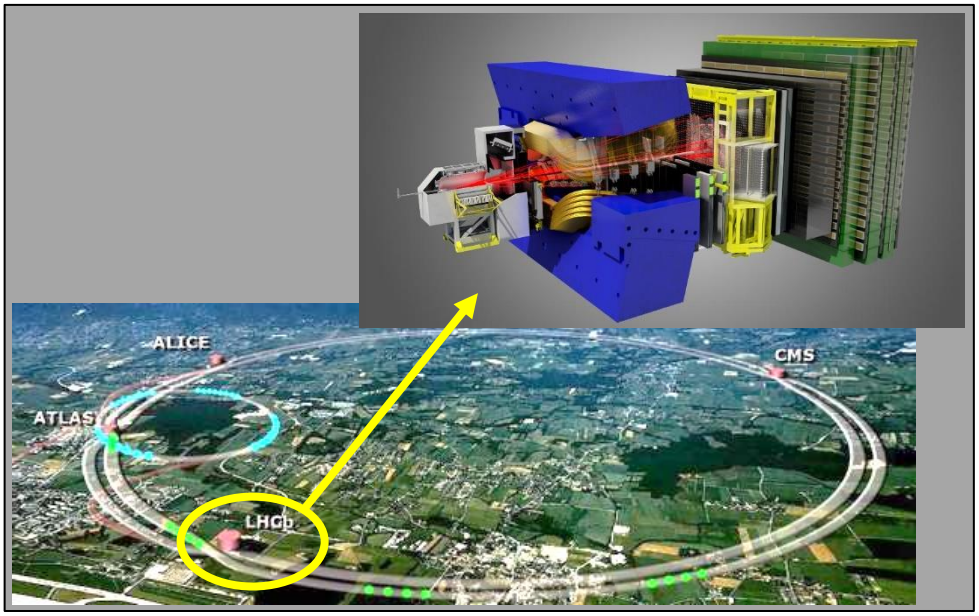


Figura 3 - Localização do experimento *LHCb*.

1.2. O Experimento *LHCb*

O *LHCb* é um experimento otimizado para realizar medidas precisas do decaimento de Mésons-B, em busca de violações da simetria CP e da descoberta de decaimentos raros além do Modelo Padrão (*SM*) [5]. Sabe-se que a produção de pares $b\bar{b}$ (objetos de estudo do *LHCb*) ocorre majoritariamente em pequenos ângulos com relação aos feixes do acelerador (ver Figura 4). Com base nesta característica, os detectores do experimento (Figura 5) foram posicionados para sensibilidade angular no intervalo $2 \leq \eta \leq 5$ (em termos de pseudo-rapidez) de modo que o experimento está otimizado para o estudo destas partículas.

Foram obtidos através do experimento *LHCb* importantes resultados no campo da física de altas energias, com 395 publicações entre 2010 e 2017 [6]. Dentre estes destacam-se, em 2012, medições de diferenças entre propriedades da matéria e da antimatéria (violações da simetria CP) nos decaimentos $B^{\pm} \rightarrow K^{\pm}\pi^{+}\pi^{-}$ e $B^{\pm} \rightarrow K^{\pm}K^{+}K^{-}$ [7] e, em 2015, a comprovação da existência dos pentaquarks através da observação de ressonâncias consistentes com sua existência no canal de decaimento $J = \psi/p$ [8].

Embora ainda não tenham sido identificados, quaisquer desvios entre medições realizadas no experimento e previsões do *SM* podem ser interpretados como pistas para a descoberta de uma nova física, além do modelo padrão.

O experimento *LHCb* é mantido por uma colaboração internacional na qual participam mais de mil membros, de 72 institutos em 16 países, e da qual o Centro Brasileiro de Pesquisas Físicas (CBPF) é membro efetivo.

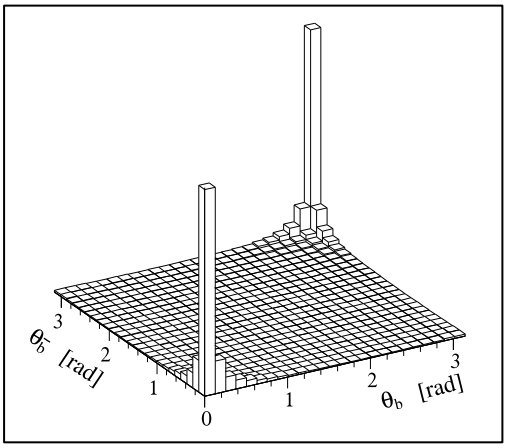


Figura 4 - Distribuição angular da produção de mésons B, calculada no simulador **PYTHIA** [9].

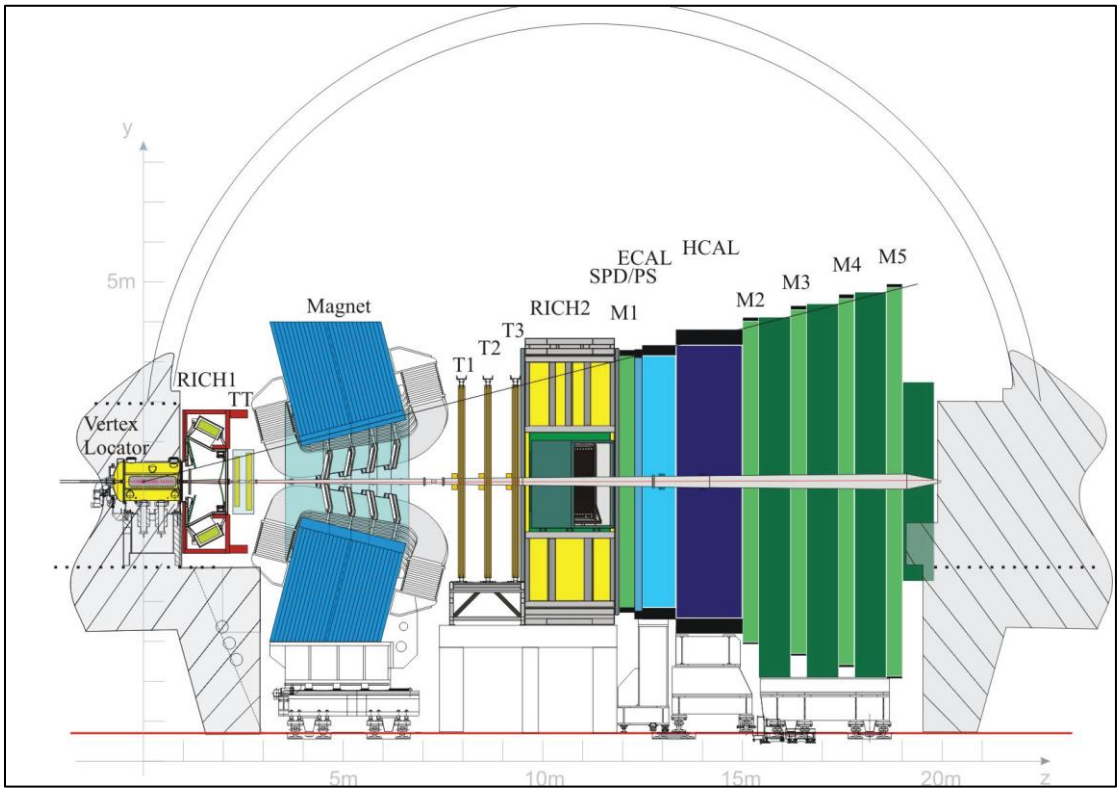


Figura 5 - Experimento *LHCb* atual [5]. O sistema de trajetografia inclui as estações *Velo*, *TT* e *T1-T3*. O sistema de identificação de partículas inclui *RICH 1* e *2*, *ECAL*, *HCAL* e *M1-M5*.

O experimento consiste, basicamente, em um sistema de trajetografia e um sistema de identificação de partículas. O sistema de identificação de partículas (*PID*) é composto por dois detectores *Ring Imaging Cherenkov Detectors* (*RICH*), para identificação de hádrons carregados, por calorímetros eletromagnético (*ECAL*) e hadrônico (*HCAL*), que permitem distinguir hádrons, elétrons e fótons e estimar suas energias, e de pelo conjunto de câmaras de múons (*M1-M5*) que identificam múons. No sistema de coordenadas[2] do *LHCb*, à montante (*upstream*) refere-se ao sentido negativo da direção z (aproximando-se do vértice primário) e à jusante (*downstream*) ao sentido positivo da mesma direção (afastando-se do vértice primário). Um grande dipolo magnético (*magnet*) produz um campo de $4\,Tm$ capaz de defletir as partículas carregadas de modo a possibilitar medidas de *momentum* através da medição da deflexão causada na partícula carregada pela força de Lorentz. Este mecanismo de medida encontra-se ilustrado na Figura 6. À montante do dipolo, o sistema de trajetografia possui um detector de tiras de silício conhecido como *Vertex Locator* (*VELO*) que determina o local da colisão pp, além de um segundo detector de tiras de silício de maior área, o *Tracker Turicensis* (*TT*). À jusante do dipolo estão três estações T (*T-stations*) que compõem o sistema de *downstream tracker* do experimento. Cada estação T é atualmente dividida em dois subdetectores que podem ser vistos na Figura 7. O *Inner Tracker* (*IT*) está na região de alto fluxo próxima ao feixe e é composto por *silicon micro-strips*, e o *Outer Tracker* (*OT*) está na região de baixo fluxo - mais afastada do feixe - e é composto por *straw drift tubes*, isto é, detectores a gás. O sistema de trajetografia será explicado com mais detalhes na Seção 1.2.1.

---

[2] O LHCb segue um sistema ortogonal de coordenadas, no qual o eixo z está definido como a direção média dos feixes do LHC, com valores positivos à medida que se afasta do ponto de interação $z = 0$. O eixo y está na vertical, e cresce para cima. O eixo x tem valores positivos na direção da caverna de acesso e negativos em direção à criogenia do LHC. Áreas positiva e negativa do eixo x são denominadas, respectivamente, "Lado A" e "Lado C" do experimento.

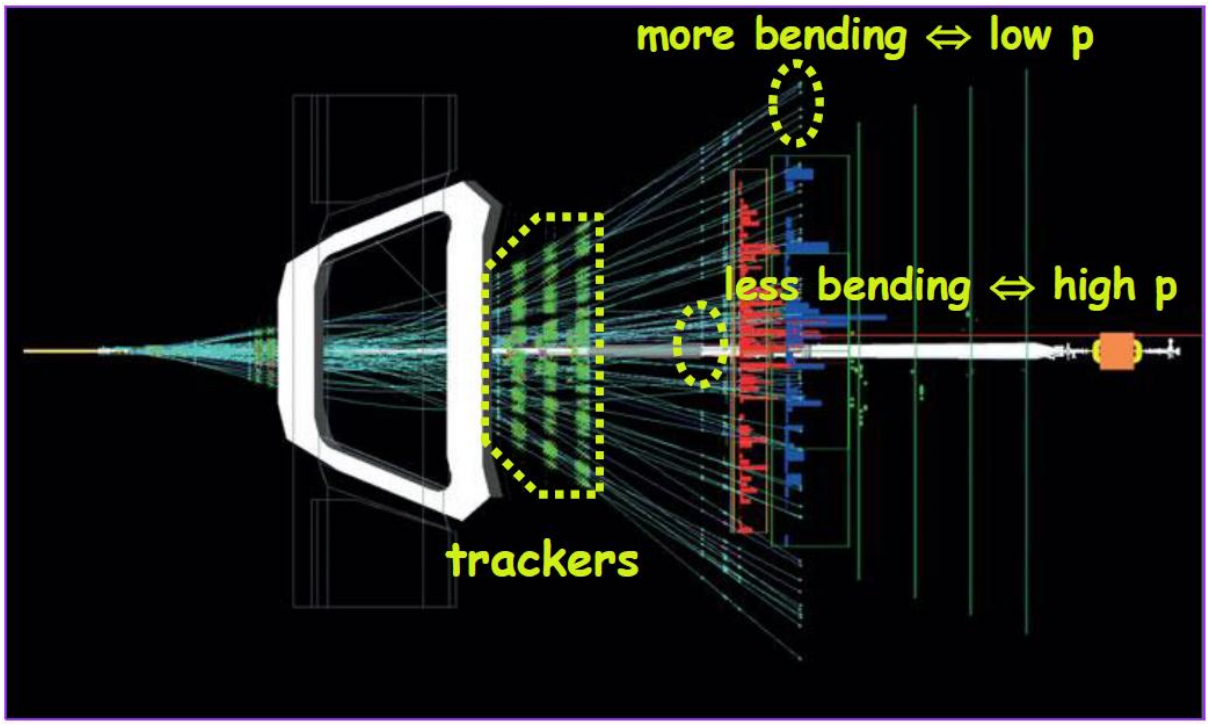


Figura 6 - Mecanismo para medida do *momentum* de partículas carregadas.

Como a quantidade de dados produzidos pelas colisões pp é muito grande para ser armazenada, um sistema de *trigger* é utilizado para selecionar estes dados. O *trigger* utilizado durante o *Run 2* está composto por dois níveis: o *trigger* do primeiro nível (L0) e o *trigger* de alto nível (HLT) [10]. O nível *L0* é processado em *hardware* e seleciona eventos a uma taxa máxima de $1{,}1\ MHz$. Posteriormente, o HLT atua em *software* reduzindo a taxa de eventos para $12{,}5\ kHz$.

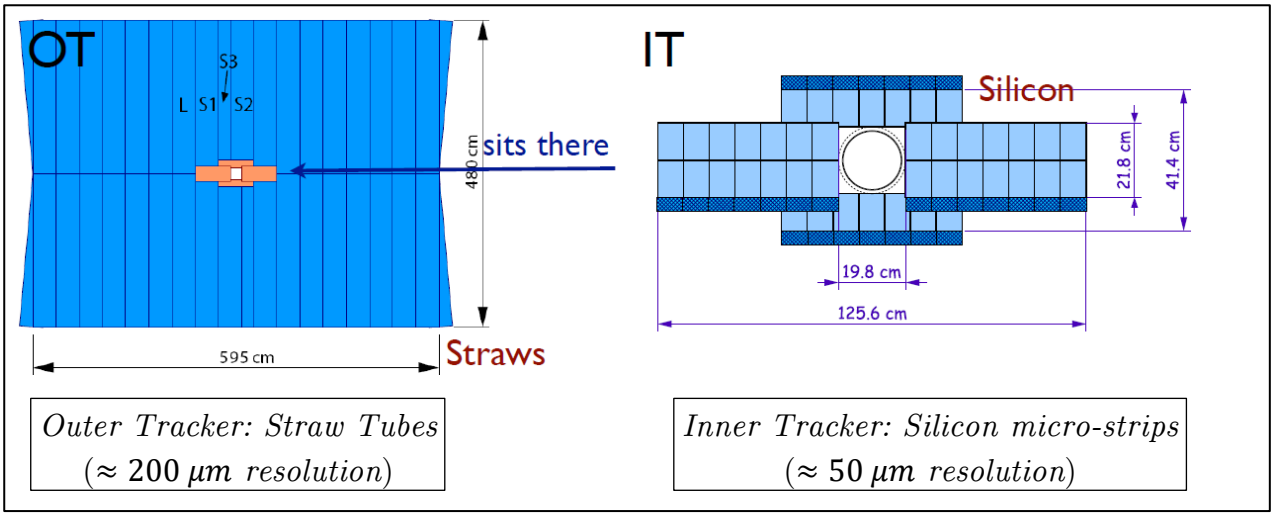


Figura 7 - Componentes das atuais estações T.

1.2.1. Reconstrução de Trajetória no *LHCb*

A reconstrução da trajetória de partículas (trajetografia) no *LHCb* é um processo computacional realizado através da combinação de dados adquiridos por diferentes detectores (*VELO*, *TT* e estações *T1*, *T2* e *T3*). Este processo começa com a composição de segmentos, obtidos através do agrupamento de pontos de passagem registrados nos detectores. Segmentos que incluem pontos do *VELO* são reconstruídos assumindo que sua origem está sobre o eixo z uma vez que o ponto da colisão pp está localizado no centro deste detector. O algoritmo procura registros de passagem em outros detectores e, se houver, eles são adicionados ao traço. Um algoritmo de busca direta procura por possíveis trajetórias para segmentos detectados no *TT*, enquanto um algoritmo de busca reversa assume que segmentos identificados nas estações *T* têm origem no ponto da colisão pp, extrapolando-os na tentativa de encontrar traços válidos.

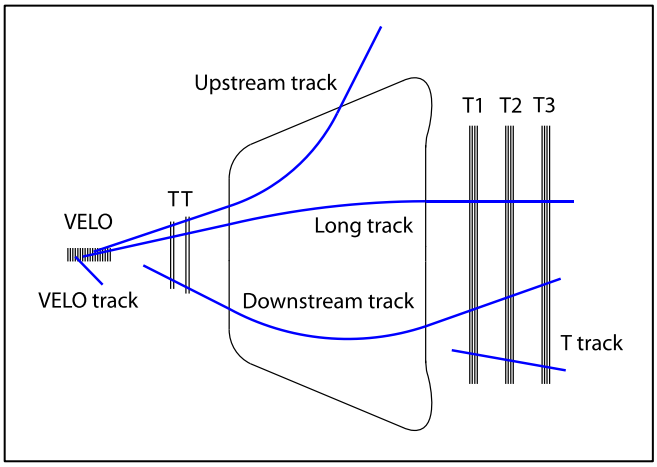


Figura 8 - Tipos de trajetórias [11].

Traços reconstruídos são rotuladas de acordo com os detectores utilizados para reconhecê-los, como mostrado na Figura 8. Traços com pontos em todos os detectores são marcados como longos (*Long Track*). Traços com pontos apenas entre *VELO* e *TT* são marcados como à montante (*Upstream Track*), traços com pontos apenas entre as estações *T* e *TT* são marcados como à jusante (*Downstream Track*) e, finalmente, traços com segmentos apenas no *VELO* ou nas estações T são marcados, respectivamente, como *VELO* (*VELO Track*) e *T* (*T-tracks*).

1.3. Motivação para a Atualização do *LHCb*

Ao longo dos primeiros anos de operação (2009 a 2013 - *Run 1*) o *LHCb* acumulou $3\,fb^{-1}$ de dados (em termos de luminosidade integrada[3]). Após a primeira grande parada de dois anos (*Long Shutdown 1* – LS1) o experimento encontra-se atualmente ativo (*Run 2*), operando com luminosidade instantânea $\mathcal{L}_{média} \approx 4 \times 10^{32}\ cm^{-2}\ s^{-1}$. Para esta luminosidade, o número médio de colisões próton-próton (pp) detectáveis por *bunch-crossing* é $\nu_{médio} = 1{,}62$.

Apesar de seu excelente desempenho, o experimento *LHCb* possui uma limitação intrínseca para aquisição de dados de cerca de $2\,fb^{-1}$ por ano [12] e a maioria dos estudos conduzidos no LHCb encontra-se atualmente limitada pela estatística disponível. Aumentar a quantidade de dados adquiridos é, portanto, fundamental para o avanço destes estudos. Tal aumento implica no incremento da luminosidade do feixe, provocando mais colisões e mais eventos por unidade de tempo. Entretanto, qualquer tentativa de operar o atual experimento LHCb com maior luminosidade leva à saturação do *trigger* L0, conforme observado na Figura 9.

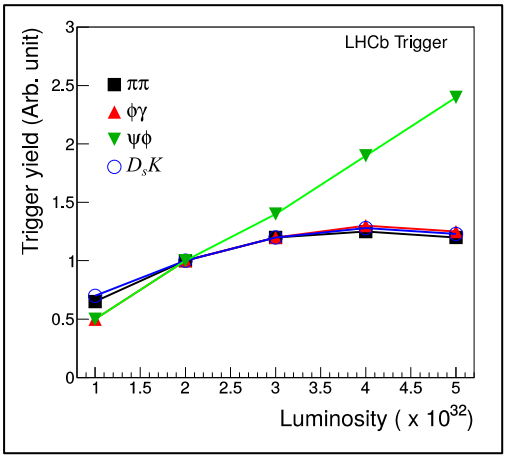


Figura 9 - Eficiência do *trigger* L0 em função da luminosidade para vários decaimentos hadrônicos [13]. Observa-se a saturação (indicada pela linha tracejada) abaixo do ponto de operação alcançado ao final do *Run 1*. O decaimento $\psi\phi$ beneficia-se de dois múons no estado final, e por isto não satura.

[3] No LHC, a luminosidade integrada do feixe é expressa em femtobarn inverso (fb^{-1}). Esta unidade representa o número de eventos de colisão de partículas por área de seção transversal do feixe.

Essa saturação é causada por limitações no *layout* dos detectores e no *hardware* de aquisição de dados, que impõem uma taxa máxima para acúmulo de dados do experimento. Como resultado, L0 deve reduzir a frequência abaixo deste limite, o que não pode ser feito de forma eficiente para luminosidades elevadas. Para aumentar a eficiência de acúmulo de dados do experimento foi necessário adotar uma abordagem completamente nova para o esquema de aquisição de dados, removendo-se o *trigger* L0 e operando o *LHCb* apenas com *trigger* baseado em *software*. A adaptação a este novo esquema exige a atualização de todos os detectores descritos na Seção 1.2.1, o que impulsionou uma completa atualização do experimento *LHCb*.

Esta atualização foi agendada para o próximo grande desligamento do *LHC*, entre 2019 e 2020. Após atualizado, o experimento será capaz de aproveitar a nova luminosidade disponível, que será 5 vezes maior que a atual. Espera-se que, desta forma, o experimento *LHCb* seja capaz de coletar $50\,fb^{-1}$ de dados ao longo do *Run 3*. O número esperado de interações pp detectáveis por *bunch-crossing* após a atualização é $v_{médio} \approx 7{,}6$.

Esta dissertação tem como foco a atualização das estações T, que após o *upgrade* passarão a ser chamadas de estações *SciFi* (*SciFi Tracker*), tema do próximo capítulo.

Capítulo 2
SciFi: Um Novo Detector Para o *LHCb*

O desenvolvimento de um dispositivo para garantia da qualidade de fabricação e caracterização da eletrônica de *front-end* contida nas caixas de leitura do projeto *SciFi* foi a motivação principal para o trabalho descrito nesta dissertação. Os principais aspectos a respeito do projeto *SciFi* serão apresentados neste capítulo.

2.1. As Atuais Estações *T*

Como introduzido na Seção 1.2.1, a instrumentação de trajetografia disposta a montante e a jusante do dipolo fornece uma medida precisa do *momentum* de partículas carregadas permitindo que se calcule a massa das partículas produzidas pela colisão pp. Trajetórias reconstruídas são também enviadas a algoritmos que as combinam com medições dos detectores *RICH*, fornecendo dados ao sistema de identificação de partículas. A atual resolução da medida de *momentum* é limitada, principalmente, pelo espalhamento múltiplo de partículas com *momentum* até cerca de $80\ GeV/c$, e pela resolução do detector para partículas com alto *momentum*.

Existem três estações de trajetografia à jusante que permitem reconhecimento de trajetória com alta eficiência e alta resolução no sentido da deflexão provocada pelo campo magnético. Cada estação é composta por quatro planos de detecção, orientados nas coordenadas $[x, u, v, x]$, respectivamente $[0°, +5°, -5°, 0°]$ em relação ao eixo vertical. Cada plano consiste em dois subdetectores, como explicado na Seção 1.2. O OT é um detector a gás de múltiplos tubos (*straw tubes*) que ocupa cerca de 99% da superfície do plano ($30\ m^2$) e o IT é um detector de micro tiras de silício que cobre uma área de apenas $0{,}35\ m^2$ ao redor do tubo de feixe (*beam pipe*). O IT apresenta resolução espacial

de cerca de $50\,\mu m$ (para eficiência mínima de 99%) [14, 15]. A resolução espacial do OT é de cerca de $200\,\mu m$ (para eficiência mínima de 99%) [16, 17]. A resolução de *momentum*, $\Delta p/p$, do sistema de trajetografia atual varia entre $0{,}4\,\%$ a $5\ GeV/c$ e $0{,}6\,\%$ a $100\ GeV/c$. Resoluções de massa invariante de cerca de $8\,MeV/c^2$ foram alcançadas para decaimentos $B \to J/\psi\, X$ com restrição aplicada à massa do J/ψ.

As atuais estações de trajetografia foram projetadas de modo a proporcionar o melhor desempenho para o atual acelerador, que produz colisões próton-próton a cada $25\ ns$ com energia de centro de massa (do projeto) $\sqrt{s} = 14\,TeV$ e luminosidade instantânea $\mathcal{L}$ entre 2 e $5 \times 10^{32}\ cm^{-1}s^{-1}$. As atuais estações de rastreamento foram concebidas para que a ocupação das regiões mais ativas do OT fosse de, no máximo, 10% com luminosidade instantânea $\mathcal{L} = 2 \times 10^{32}\ cm^{-1}s^{-1}$ [11]. Após algumas melhorias nos algoritmos de reconstrução de trajetórias foi possível coletar dados com luminosidade instantânea $\mathcal{L} = 5 \times 10^{32}\ cm^{-1}s^{-1}$. Neste caso, a ocupação no OT aumentou para 25% nas regiões mais ativas sem perda de eficiência.

2.2. Motivação para o Projeto *SciFi*

A atualização do LHCb objetiva aproveitar o incremento de luminosidade que haverá após a atualização do LHC, quando a luminosidade instantânea $\mathcal{L}$ estará entre 1 e $2 \times 10^{33}\ cm^{-1}s^{-1}$. Esta luminosidade será obtida com o mesmo intervalo atual de $25\ ns$ $(40\ MHz)$ entre *bunch-crossings*, resultando numa quantidade média de interações próton-próton detectáveis por *bunch-crossing* $\nu_{média}$ entre $3{,}8$ e $7{,}6$. Foi mostrado [12] que a luminosidade disponível a partir da atualização do LHC provocará uma ocupação das regiões mais ativas do OT muito grande para que os algoritmos de reconstrução de trajetórias funcionem de forma eficiente e que, portanto, o OT precisaria ser substituído. Além disto, a coleta de dados nesta luminosidade só é possível se o atual *trigger* de *hardware* (L0) for removido, o que significa que o experimento deve

ser lido à taxa plena de $40MHz$. O *hardware* de leitura, tanto do IT quanto o do OT, é incapaz de operar nesta frequência e deve, portanto, ser substituído [11].

Por fim, considerou-se a grande quantidade de modificações necessárias, a complexidade[4] da substituição do *hardware* de leitura do IT e a facilidade do uso de uma tecnologia de detecção comum e se decidiu substituir por completo as estações T.

2.3. Principais Desafios do Projeto *SciFi*

Uma boa reconstrução da trajetória exige um detector com alta eficiência, alta resolução espacial no plano de deflexão do dipolo e baixa densidade de material em sua área sensível. Como dito, a atualização do *LHCb* exigirá também que o sistema de aquisição funcione à frequência plena de $40\ MHz$. O detector deverá ser capaz de operar ao longo de toda a vida útil prevista para o experimento *LHCb* atualizado sem perda expressiva de rendimento, garantindo desempenho de reconstrução da trajetória bom o suficiente durante toda a execução do experimento.

2.3.1. Performance

Em relação à performance do detector, os principais requisitos levados em conta no projeto das estações *SciFi* foram:

- Maior eficiência possível para detecção da passagem de partículas ($\approx 99\%$) mantendo taxa de ruído baixa ($< 10\%$) para reconstrução de clusters em qualquer local do detector (em relação à taxa de sinal na mesma localização);

[4] A localização central do IT requer que seu *hardware* de leitura resista a elevadas doses de radiação ionizante devido à proximidade do feixe. Além disto, o *design* do *hardware* de leitura deve levar em conta efeitos de espalhamento múltiplo provocados pelo material introduzido.

- Resolução espacial no plano de deflexão do dipolo inferior ou igual a $100\mu m$. Não é necessária uma resolução melhor pois a extrapolação de faixas do VELO é dominada pelo espalhamento múltiplo nos detectores à montante do dipolo.

- Mínima densidade de material na região sensível, de tal forma que os efeitos de espalhamento múltiplo provocados pelo detector sejam menores do que aqueles causados pelo material à montante do dipolo.

- Eletrônica de leitura capaz de funcionar a frequência plena de $40MHz$ e tempo de recuperação dos canais de leitura pequeno o suficiente para mitigar a ineficiência devida ao "tempo morto".

- Estações devem ser capazes de operar com o desempenho necessário até uma luminosidade integrada de, pelo menos, $50\ fb^{-1}$.

2.3.2. Geometria

O projeto *SciFi* é composto por três estações colocadas em posições equidistantes ao longo da linha do feixe, entre o dipolo e o calorímetro RICH2. O espaço disponível para estas estações é limitado ao usado atualmente pelas estações T. Toda a infraestrutura de suporte atual deve ser reutilizada.

Outro aspecto importante é que o ponto de passagem de uma partícula que atravessa o detector precisa ser muito bem determinado. Para isto as fibras devem estar retas, respeitando uma tolerância máxima de $\pm 100\ \mu m$ na direção x, e os planos devem ser posicionados com tolerância de $\pm 300\ \mu m$ na direção z, garantindo desta forma que os requisitos de resolução sejam satisfeitos. As estações precisarão, portanto, ser minunciosamente alinhadas após sua montagem.

2.3.3. Operação em Ambiente Radioativo

O nível de exposição das estações *SciFi* à radiação ionizante foi objeto de um estudo à parte, que utilizou o *software* de simulação **FLUKA** [15, 16]. A simulação levou em conta a versão mais recente da geometria dos detectores e as condições de feixe esperadas após a atualização, ou seja, $\sqrt{s} = 14\,TeV$, luminosidade integrada de $50\,fb^{-1}$ e seção de choque total das colisões próton-próton (pp) igual a $100\,mb$. Maiores detalhes deste estudo podem ser obtidos na Ref. [18].

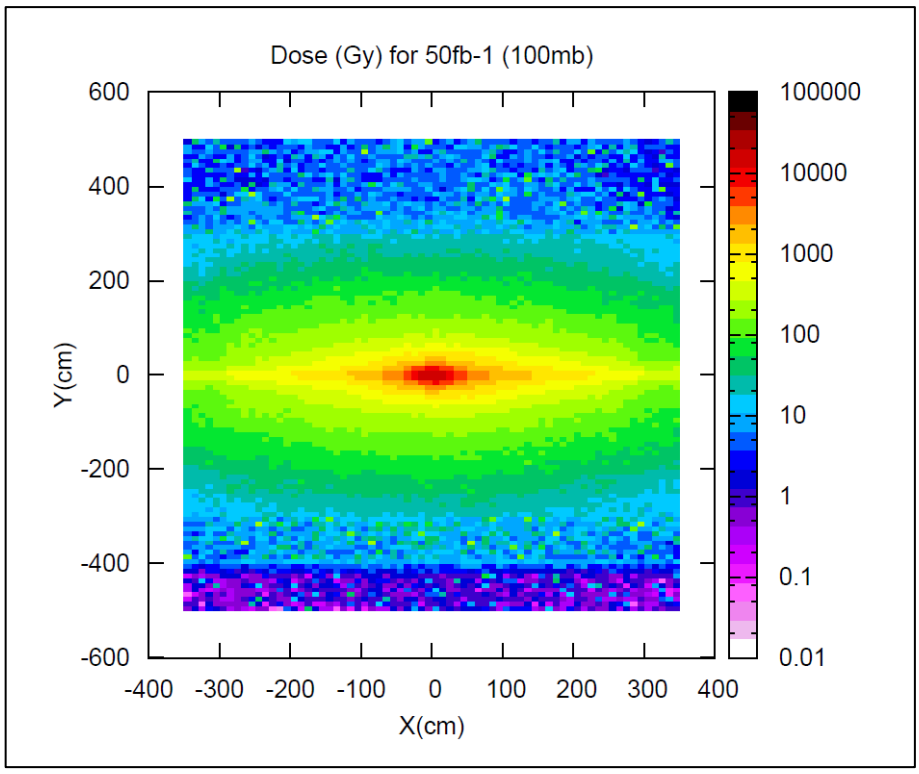


Figura 10 - Expectativa de dose acumulada no plano $x - y$, em $z = 783\,cm$, ao longo da vida útil prevista para o detector ($50\,fb^{-1}$ de luminosidade integrada) [18].

A expectativa para a dose acumulada ao longo da vida útil do detector é mostrada na Figura 10. A fluência de nêutrons máxima para a estação T3 na posição dos *SiPM's* ($y = \pm 250\,cm$) após $50\,fb^{-1}$ é estimada em $13 \times 10^{11}\,n_{eq}/cm^2$, representando uma dose integrada de $80\,Gy$. Espera-se que a dose máxima absorvida nas fibras cintilantes seja de cerca de $35\,kGy$, em T1. As simulações também mostraram que uma blindagem tecnicamente viável, construída a partir de polietileno com 5% de boro, pode reduzir a fluência de nêutrons na região dos *SiPM's* pela metade.

A eletrônica próxima ao feixe deve sobreviver a este ambiente. Este importante aspecto do projeto levou ao desenvolvimento de uma série de dispositivos aptos a operar em ambiente radioativo, como os chips **GBTx**, **GBT-SCA** e **PACIFIC**, conversores DC-DC de tensão e adaptadores ópticos.

A dose acumulada resultará ainda em incremento da corrente de escuro das *SiPM's* e comprometimento da transparência das fibras ópticas. Para controlar estes efeitos indesejados foram adotadas algumas medidas: Resfriamento das *SiPM's* (descrito a seguir); Clusterização (descrito em 2.8.3); Otimização das *SiPM's* para redução de *cross-talk* e de *after pulses*; Inclusão de um *shaper* no *readout* analógico (descrito em 2.8.2) e instalação da blindagem de nêutrons na região dos *SiPM's*.

2.3.4. Resfriamento

Sabe-se que danos por radiação degradam o desempenho do detector à medida que aumentam a corrente de escuro das *SiPM's*, afetando seu pedestal de ruído. Embora *SiPM's* operem em temperatura ambiente, danos causados por radiação podem ser mitigados reduzindo-se a agitação térmica do material semicondutor – com consequente redução da corrente de escuro – através de seu resfriamento. A dependência de parâmetros elétricos exige resfriamento estável e homogêneo, com a temperatura variando não mais que $\pm 1K$ ao longo dos *arrays* de *SiPM's*. Isto será obtido pela circulação de fluido refrigerante através de trocadores de calor acoplados às fotomultiplicadoras.

2.4. As Novas Estações *SciFi*

O projeto das novas estações do *Scintillating Fiber Tracker* (*SciFi Tracker* ou SFT) foi baseado na tecnologia de fibras ópticas cintilantes, lidas por fotomultiplicadores de silício (*SiPM's*) conectadas às caixas de leitura (*Read-out Boxes*) nas partes superior e inferior das estações. Este *design* é similar ao demonstrado na referência [19].

As estações de trajetografia *SciFi* estarão posicionadas onde atualmente se encontram as três estações T (T1, T2, T3), conforme representado na Figura 11.

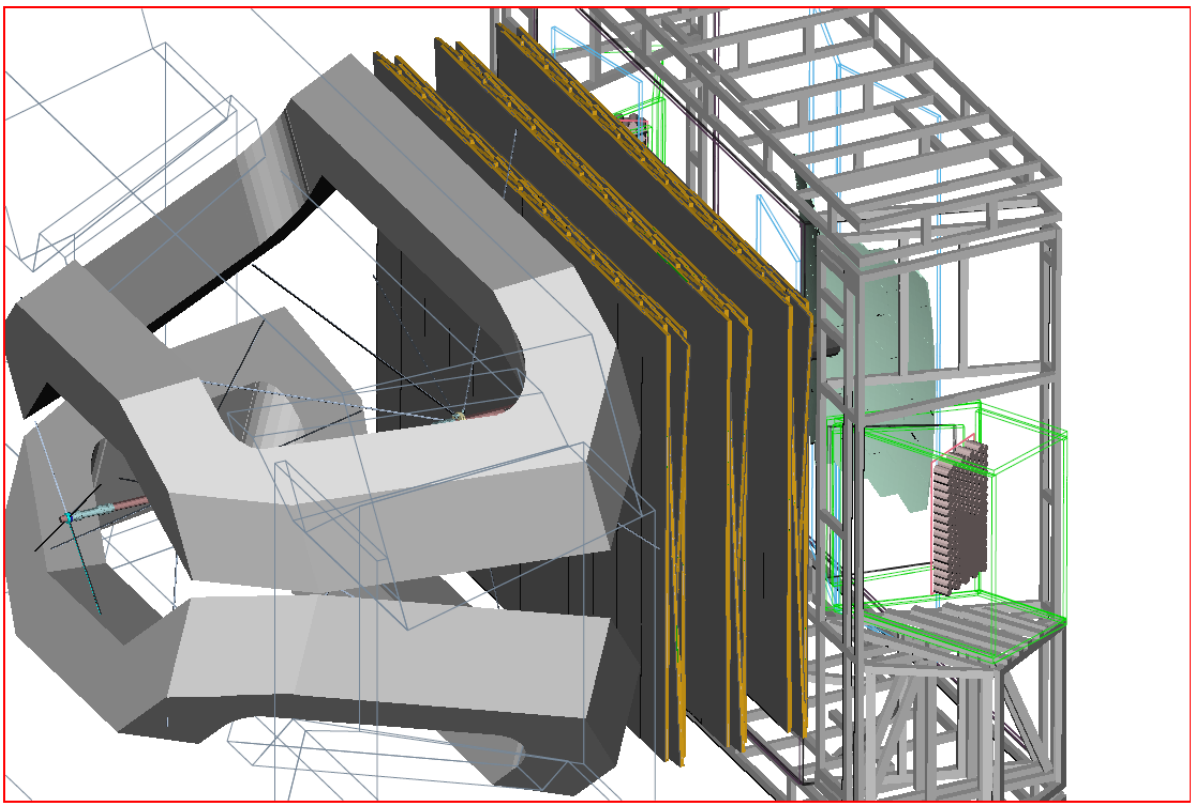


Figura 11 - As três estações de fibras cintilantes entre o dipolo (à esquerda) e RICH2 (à direita) [11].

Cada estação *SciFi*, representada na Figura 12, será composta por quatro planos de detecção. Cada plano possui superfície ativa composta por $\approx 254 \times 10^3$ fibras com $250\ \mu m$ de diâmetro e $2{,}5\ m$ de comprimento. A distância entre planos de uma estação é de $20\ mm$ ao longo do eixo z. Haverá um orifício central circular para viabilizar a passagem do tubo de feixe do LHC. Em cada estação os planos externos estarão alinhados em relação ao eixo y ("*x-layers*"), enquanto que os planos internos estarão inclinados em $\pm 5°$ em relação ao eixo y ("*u-*" e "*v-layers*"). Este *design* permite a identificação do ponto da passagem das partículas no eixo y através do efeito *stereo*, como demonstrado na Figura 13. Cada plano foi subdividido em 12 módulos detectores, como mostrado na Figura 14. Serão, portanto, necessários $12 \times 4 \times 3 = 144$ módulos para a montagem das três estações.

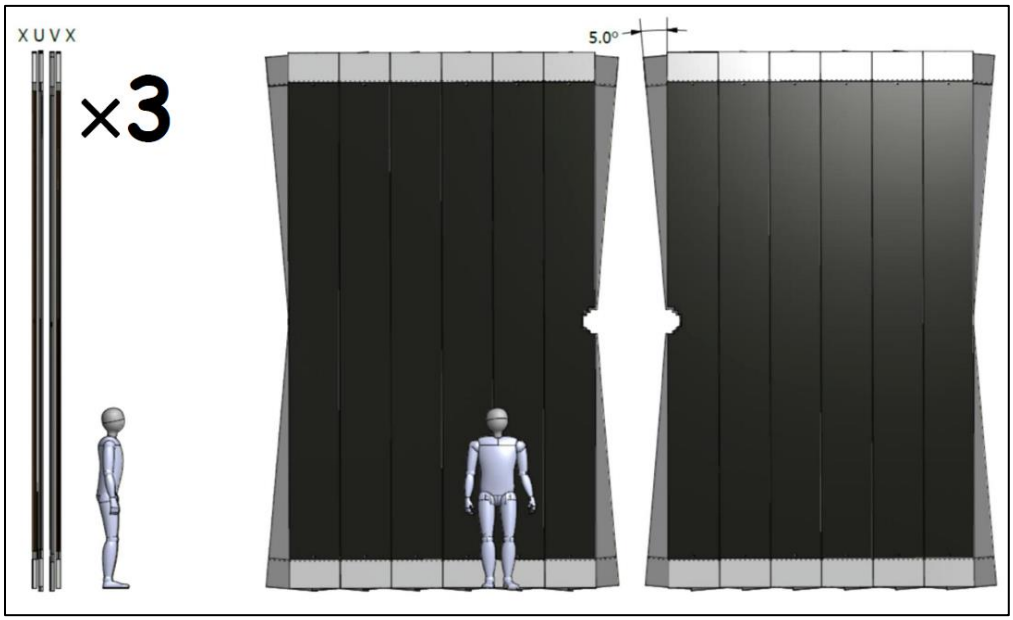


Figura 12 - Representação em vistas de uma estação SciFi.

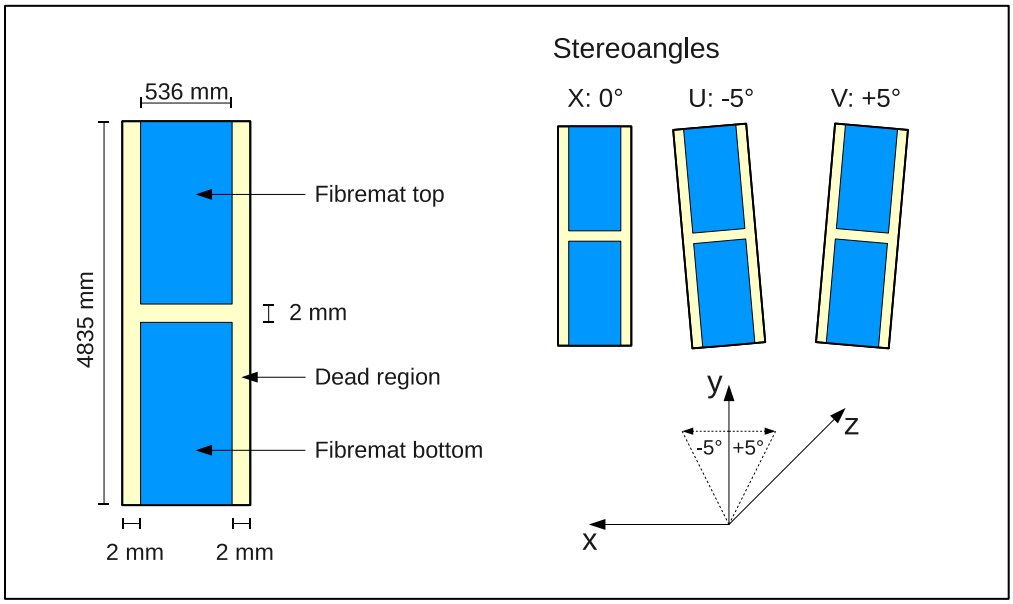


Figura 13 – Dimensões de um Módulo e definição dos ângulos *stereo* [11].

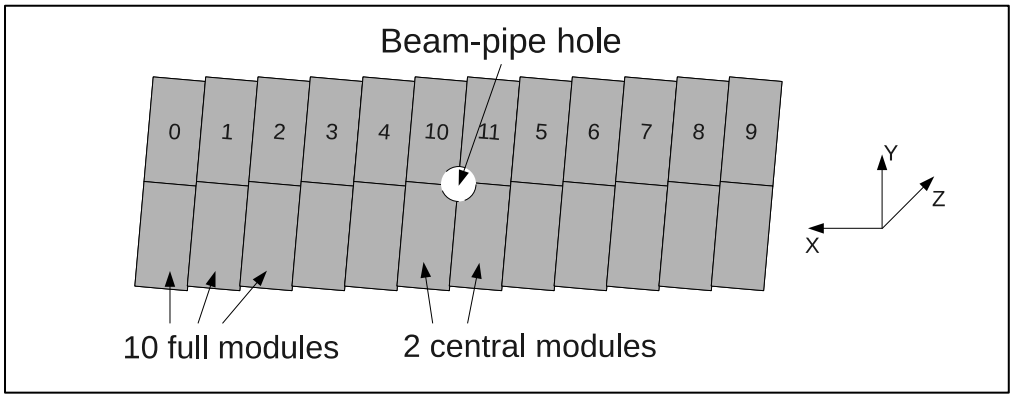


Figura 14 – Estrutura de um plano de detecção composto por 12 módulos [11].

2.5. Fibras Ópticas Cintilantes

Fibras plásticas cintilantes são os elementos detectores do projeto *SciFi* e têm duas funções principais: (1) converter em fótons a excitação atômica provocada por partículas carregadas que as atravessam e (2) condicionar e transportar o sinal óptico para os dispositivos de leitura, localizados fora da área sensível.

A fibra óptica cintilante utilizada no projeto *SciFi* possui um núcleo de PMMA dopado com poliestireno (índice de refração $\eta_1 = 1{,}59$), uma primeira camada de revestimento (casca) produzida com PMMA ($\eta_2 = 1{,}49$) e uma segunda casca ($\eta_3 = 1{,}42$) fabricada a partir de um polímero fluorado. A espessura de cada casca é de cerca de 3% do diâmetro total ($250\,\mu m$) da fibra. A Figura 15 ilustra esta estrutura.

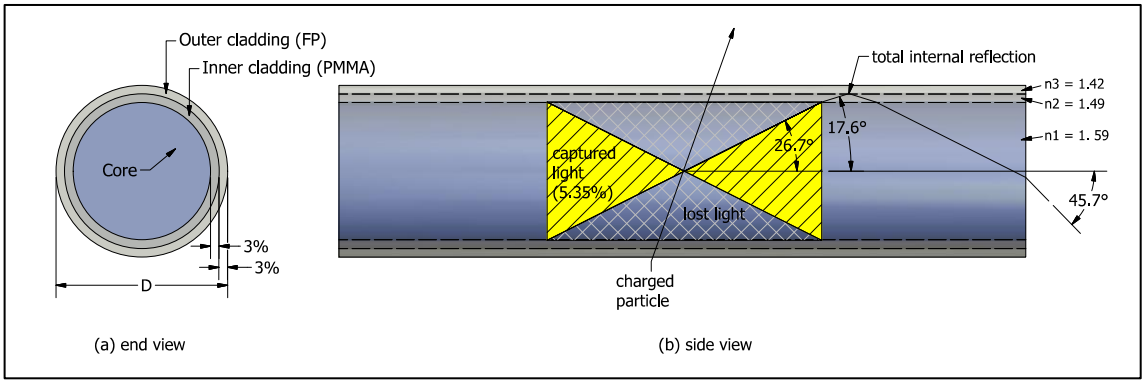


Figura 15 - Representação da estrutura de uma fibra cintilante [20].

A geração de sinais através da fibra plástica cintilante é um processo de interações sucessivas. A energia de ionização depositada no núcleo da fibra por uma partícula que o atravessa provoca a excitação em nível molecular (especificamente, dos chamados elétrons π) do polímero. O tempo de relaxamento e o rendimento de cintilação luminosa do poliestireno são, porém, pobres. Um dopante fluorescente orgânico ativador com nível de energia de excitação compatível com a ionização depositada pela partícula a ser detectada é então adicionado à base de poliestireno ($\approx 1\%$ em peso) para melhorar a eficiência do mecanismo de cintilação. A energia é transferida rapidamente ($< 1\,ns$) da base para este dopante 'ativador' por meio de uma transmissão dipolo-dipolo não-radiativa conhecida como transferência de energia Förster ressonante, em que o estado

de energia excitado do dopante será, posteriormente, relaxado pela emissão de um fóton. O dopante ativador é escolhido para que se obtenha elevado rendimento quântico ($> 95\%$), emissão em comprimento de onda específico, e rápido decaimento (menor que alguns ns). Entretanto, o dopante ativador emite na região do azul fazendo com que esta região acabe sofrendo forte atenuação devido a auto-absorção de luz pelo próprio dopante. Por esta razão é adicionado um segundo dopante no núcleo ($\approx 0{,}05\%$ em peso) que absorve fótons emitidos pelo ativador em azul ou UV reemitindo-os isotopicamente em comprimentos de onda mais longos (azul-esverdeado). Neste intervalo de comprimento de onda a probabilidade de reabsorção pelos dopantes é reduzida e se obtêm melhor transparência do poliestireno.

O transporte da luz na fibra baseia-se na reflexão interna total causada pela interface entre o núcleo de fibra e suas cascas. Raios meridionais, ou seja, aqueles que cruzam o eixo de fibra, são refletidos se o ângulo de incidência em relação à normal da superfície exceder o ângulo crítico definido por:

$$\theta_{crítico} = \sin^{-1}\left(\eta_{casca}/\eta_{núcleo}\right) \quad (1)$$

A inclusão do segundo revestimento externo fluorado ($\eta_3 = 1{,}42$) permite aumentar o aproveitamento de fótons gerados em raios meridionais de $3{,}1\%$ para $5{,}35\%$, como demonstrado na Figura 15.

Uma estimativa da quantidade de fótons disponíveis em uma das extremidades da fibra quando uma partícula de mínima ionização (*Minimum Ionization Particle*) a atravessa cruzando seu eixo ($dx = 250\,\mu m$) pode ser obtida com os parâmetros disponíveis na referência [21] da seguinte forma:

$$\text{MIP} = dE/dx = 2\,[MeV/cm]$$

$$\text{Energia Depositada } (dE) = MIP \times dx = 2\,[MeV/cm] \times 0{,}025\,[cm] = 0{,}05\,[MeV] \quad (2)$$

$$\text{Rendimento de Cintilação} = 8000\,[\gamma\ /\ MeV]$$

$$\text{Fótons Produzidos} = 0{,}05\,[MeV] \times 8000\,[\gamma/MeV] = 400\,[\gamma] \quad (3)$$

$$\text{Fótons Aprisionados na Fibra} = 400[\,\gamma] \times 5{,}35\% \approx 21\,[\gamma] \quad (4)$$

$$\text{Perda por Atenuação em 1 } [m] \text{ de Fibra} \approx 26\,\%$$

$$\text{Fótons Efetivamente Disponíveis} \approx 21\,[\gamma] \times (1 - 26\,\%) \approx \mathbf{16}\,[\boldsymbol{\gamma}] \quad (5)$$

Considerando ainda as perdas adicionais devidas à eficiência da fotomultiplicadora ($PDE_{typ} \approx 25\%$) fica evidente que a quantidade efetivamente disponível de fótons é bastante pequena. Para aumentar então a eficiência do detector cada plano está sendo construído a partir da sobreposição de 5 ou 6 camadas[5] de fibras ópticas cintilantes, resultando numa seção transversal como a que é vista na Figura 16.

Figura 16 - Plano de fibras cintilantes demonstrado por *B. Beischer et al* [19].

[5] Módulos mais próximos ao tubo de feixe receberão uma camada adicional de fibras cintilantes para aumentar seu rendimento e, assim, compensar os danos por radiação ao longo de sua vida útil.

O projeto de montagem dos planos detectores pode ser visto na Figura 17 e um protótipo construído para o projeto SciFi é mostrado na Figura 18.

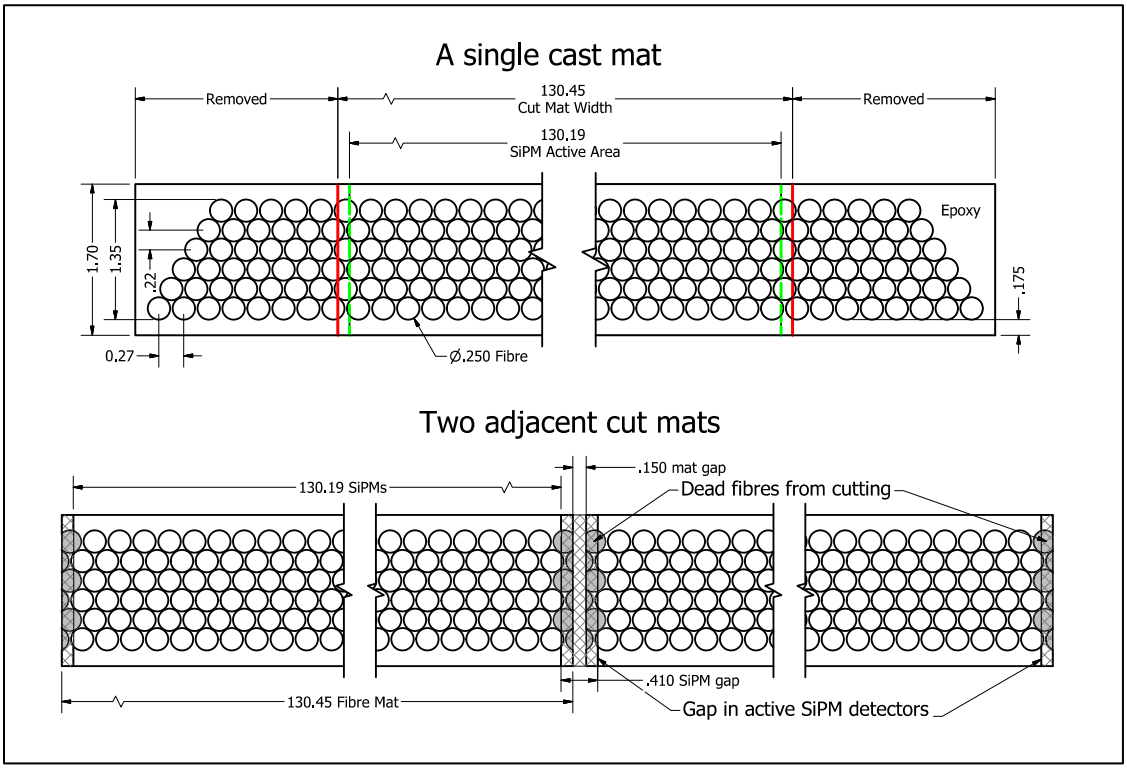


Figura 17 – Vista em corte transversal dos planos de detecção [11].

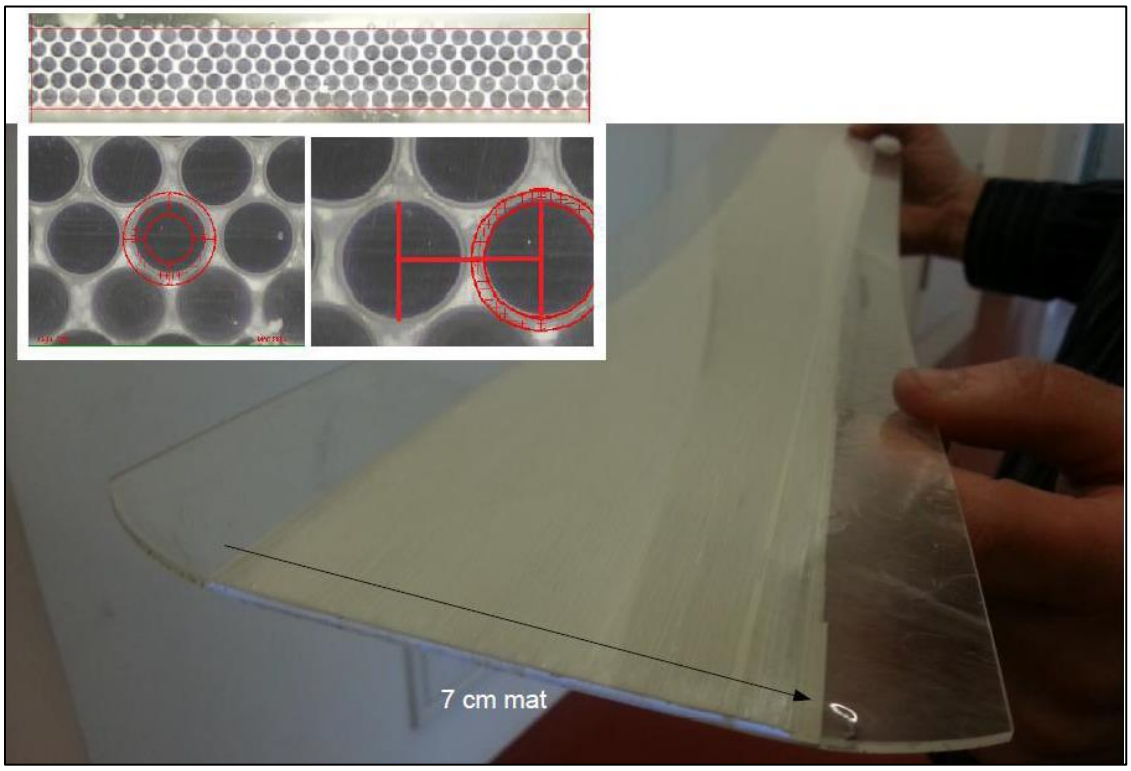


Figura 18 - Protótipo do plano detector das estações *SciFi* [11].

2.6. Fotomultiplicadoras de Silício (*SiPM's*)

Fotomultiplicadoras do tipo *SiPM* são fotodetectores a base de silício que apresentam rápida resposta, rápida regeneração, tamanho compacto e boa sensibilidade a fótons unitários. Estas características as tornam ideais para aplicação nas estações de trajetografia do projeto *SciFi*. Uma visão geral das características destes dispositivos pode ser encontrada nas referências [22, 23, 24, 25, 26, 27, 28, 29].

Um canal *SiPM* é composto por múltiplos diodos detectores de fótons por avalanche (*APD's - Avalanche Photo-Diodes*) operados em modo Geiger-Müller e dispostos numa área da ordem de mm^2 na forma de *pixels* conectados em paralelo. A Figura 19 detalha as características geométricas das *SiPM's* utilizadas no *SciFi*.

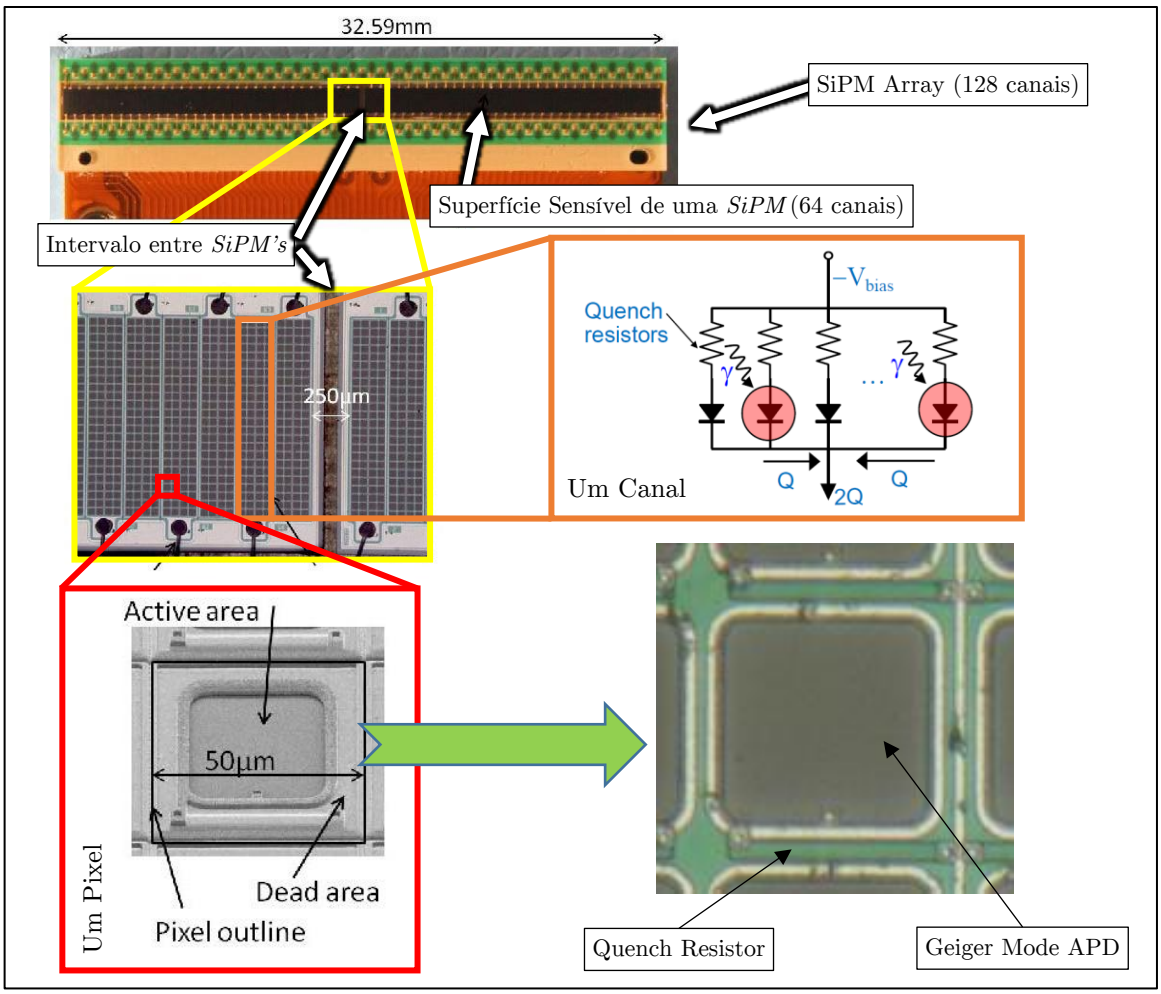


Figura 19 - Detalhes da construção da fotomultiplicadora de silício utilizada no *SciFi*.

Como cada pixel é operado em modo Geiger-Müller mesmo se apenas um único fóton for absorvido por determinado pixel será nele desencadeada uma descarga completa (avalanche), com liberação total de carga. Quando a descarga se encerra a corrente conduzida por um resistor de dreno faz com que a célula se recarregue. A quantidade de carga produzida por pixel disparado corresponde ao ganho de foto multiplicação de forma que $Q = G \cdot e$, onde G é o ganho, Q é a carga produzida e e é a carga elementar do elétron.

O ganho depende da área de superfície do pixel, da espessura da região de amplificação e da tensão de operação acima do limite de avalanche, além de ser variante em função do comprimento de onda do fóton incidente. Diferentes *SiPM's* disponíveis no mercado proporcionam ganhos da ordem de 10^6 ou 10^7 para pixels de $50 \times 50\,\mu m^2$, por exemplo.

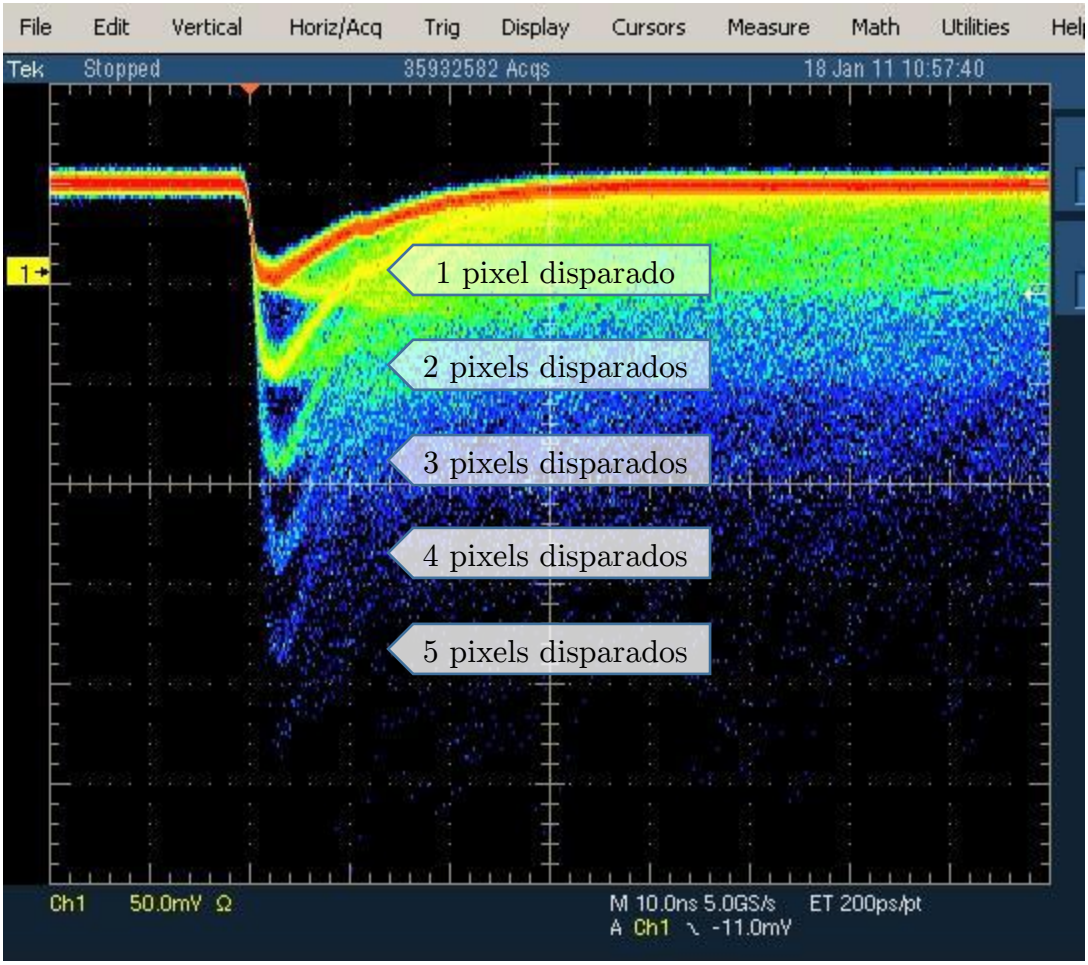


Figura 20 - Típico sinal de saída de uma fotomultiplicadora de silício.

O sinal total produzido por um canal num sensor *SiPM* será a soma da carga de todos os pixels disparados simultaneamente. Este efeito é demonstrado na Figura 20, onde um sinal de saída de uma *SiPM* típica foi capturado por um osciloscópio em modo de persistência estatística.

Impurezas e geração térmica de cargas livres são responsáveis por uma taxa permanente de avalanches não induzidas por fótons. Isso resulta em um ruído de contagem no escuro (f_{DCR}, ou *Dark Count Rate*) da ordem de várias centenas de kHz/mm^2 em temperatura ambiente para uma *SiPM* típica. Levando em conta a necessidade de detecção de fotoelétrons unitários (PE, em inglês *Photoelectrons*) nota-se que o sinal produzido pelo ruído no escuro será - a priori - indistinguível do sinal produzido por fótons unitários caso o limiar de detecção esteja definido para, por exemplo, $0{,}5\,PE$.

Durante a última década fabricantes de *SiPM's* fizeram progressos significativos na redução da DCR através de otimizações no material e no *design*. Atualmente, os melhores dispositivos apresentam em temperatura ambiente $f_{DCR} \approx 100\,kHz/mm^2$. No entanto, tal como em outros dispositivos de silício, esta performance é afetada pela exposição a nêutrons atingindo rapidamente níveis onde f_{DCR} torna-se insustentável. No projeto *SciFi* a sensibilidade para detecção de fótons unitários precisa ser preservada e a f_{DCR} precisa ser controlada através da temperatura operacional, que tipicamente reduz f_{DCR} pela metade a cada $10\,K$ de resfriamento.

Outras fontes de ruído são os pulsos fantasmas causados por cargas acumuladas perto da região de avalanche. Estes "pós-pulsos" ocorrem atrasados em relação à avalanche primária, ou seja, num momento em que o pixel ainda está recarregando. Sua amplitude depende deste atraso e geralmente é menor do que um pulso induzido por fótons. Outro efeito muito significativo para a aplicação no projeto *SciFi* é o *cross-talk* óptico que ocorre quando, durante a avalanche num pixel um grande número de fótons é produzido na região do IR (infravermelho). Como o silício é, em grande parte, transparente para o IR estes fótons atingem pixels vizinhos desencadeando neles avalanches

secundárias sincronizadas com a avalanche primária. Efeitos adversos causados por estas duas últimas fontes de ruído são mitigados através do processo de clusterização.

2.7. Mecanismo de Leitura

O mecanismo de leitura adotado nas estações do projeto SciFi está baseado no acoplamento óptico de uma das extremidades do plano detector de fibras cintilantes a diversas fotomultiplicadoras *SiPM* multicanais, como visto na Figura 21.

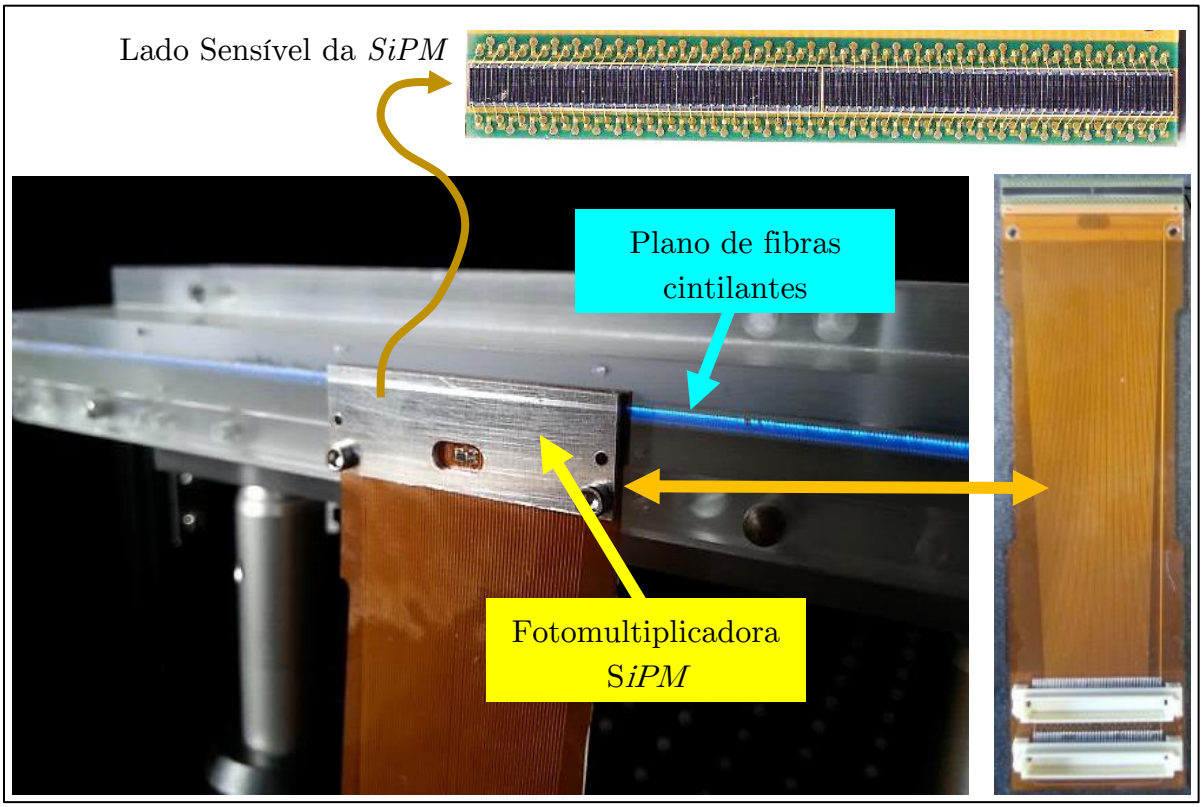


Figura 21 - Acoplamento do plano de fibras cintilantes na *SiPM*.

Fótons produzidos pela passagem de partículas carregadas são conduzidos até a extremidade do plano de fibras cintilantes e, então, para as fotomultiplicadoras. A leitura ocorre através do acoplamento ótico da seção reta deste plano detector aos *pixels* de diversos canais das *SiPM's*. A Figura 22 ilustra este mecanismo.

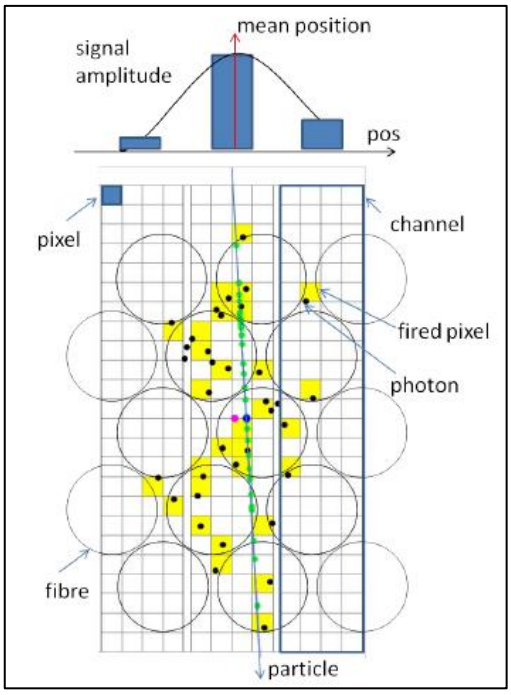


Figura 22 - Leitura do ponto de passagem de uma partícula pelo plano de fibras cintilantes [11].

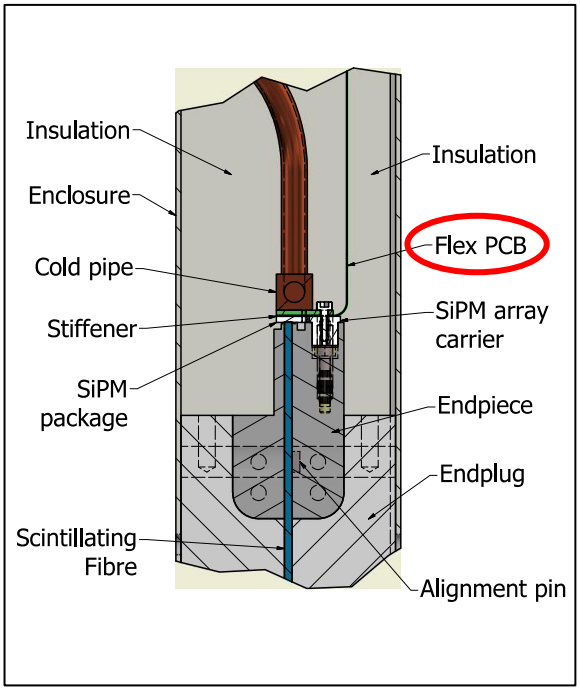


Figura 23 - Detalhe das conexões entre a caixa de leitura e o plano de fibras cintilantes [11].

Em cada pixel da fotomultiplicadora o efeito de avalanche é provocado por fótons incidentes. A quantidade de carga do sinal produzido por cada canal, representada pelas barras da parte superior da figura, será proporcional à quantidade de pixels disparados (em amarelo) no respectivo canal. Observe que as fibras não precisam estar alinhadas aos canais das fotomultiplicadoras, não sendo este parâmetro crítico.

A posição de passagem de uma partícula é obtida através de um algoritmo de clusterização a partir do cálculo do centro de massa do histograma observado. Esta técnica objetiva incrementar a resolução espacial do detector ao mesmo tempo em que reduz o volume de dados[6] a ser transmitido para o *back-end* do experimento [30].

As matrizes *SiPM* devem estar conectadas aos circuitos eletrônicos de *front-end*[7] para que seja possível a leitura dos sinais elétricos. Esta conexão se dá através de cabos PCB flexíveis, como mostrado na Figura 23.

---

[6] 1.024 linhas seriais a $160\ Mbps$, i.e. $\approx 164\ Gbps$ por Caixa de Leitura (sem compressão).

[7] O termo *front-end* se refere ao conjunto de elementos do sistema de *data acquisition* que se encontra localizado na vizinhança imediata dos detectores.

2.8. Caixas de Leitura (*Read-out Boxes*)

Localizadas nas extremidades superior e inferior de cada módulo detector estas caixas de volume relativamente pequeno possuem alta densidade de interfaces e são designadas como “Caixa de Leitura" (ROB, ou *Read-out Box*). Seu *design* pode ser visto na Figura 24.

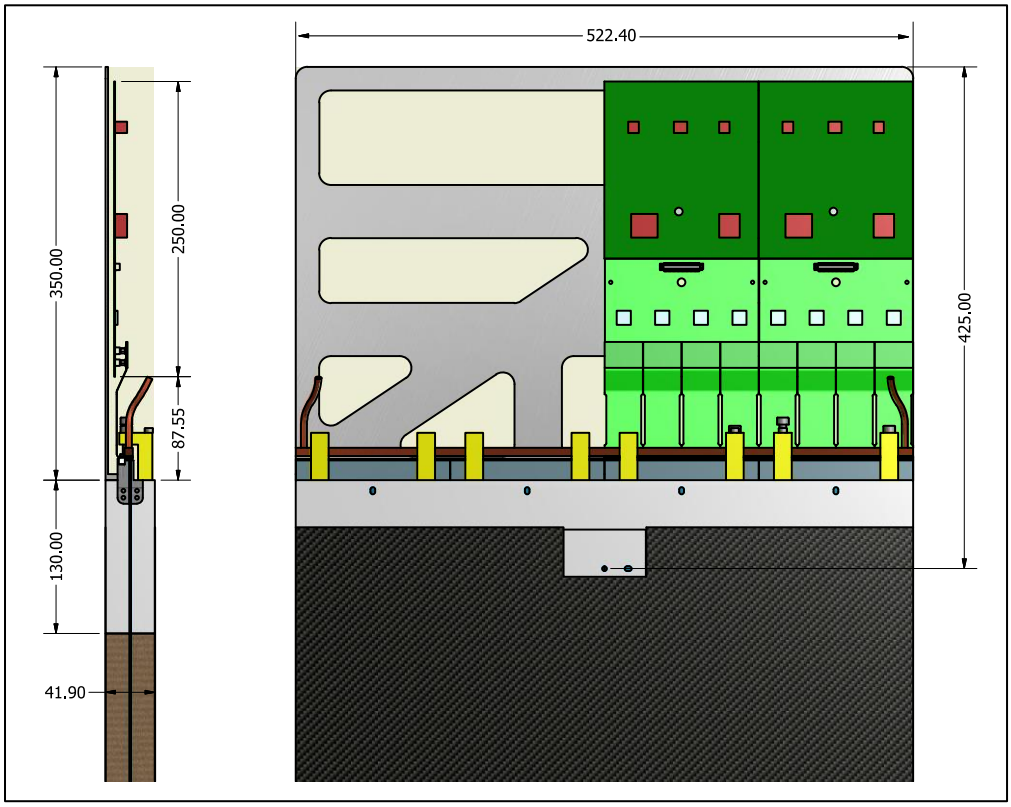


Figura 24 - *Design* da caixa de leitura [11]

O *design* da caixa de leitura é tecnicamente desafiante não só devido ao espaço limitado e à grande densidade de interfaces de refrigeração, eletrônicas e mecânicas, mas também devido aos requisitos de que a caixa de leitura seja leve, com interior termicamente bem isolado e estanque. Para evitar a condensação e o acúmulo de gelo nas superfícies frias do interior da caixa de leitura será utilizado um gás seco para preencher todo seu volume interno. Este gás deve apresentar ponto de orvalho abaixo de $-70ºC$ na entrada e abaixo de $-50ºC$ no retorno.

O interior da caixa de leitura será acessível através de dois painéis na face frontal da caixa, um para os dispositivos da eletrônica *front-end* e outro para o volume frio onde estão os *SiPM's*. O acesso para substituição de partes da eletrônica, *SiPM's* e para reparo de conexões deve ser possível mesmo quando o módulo já estiver na posição fechada, no interior do experimento. As fotomultiplicadoras *SiPM* foram agrupadas em *super-arrays*[8] permitindo sua fácil substituição, caso necessário. Os conectores de fibra óptica e de energia estão na face superior da caixa. Uma ilustração da aparência final destas caixas é mostrada na Figura 25.

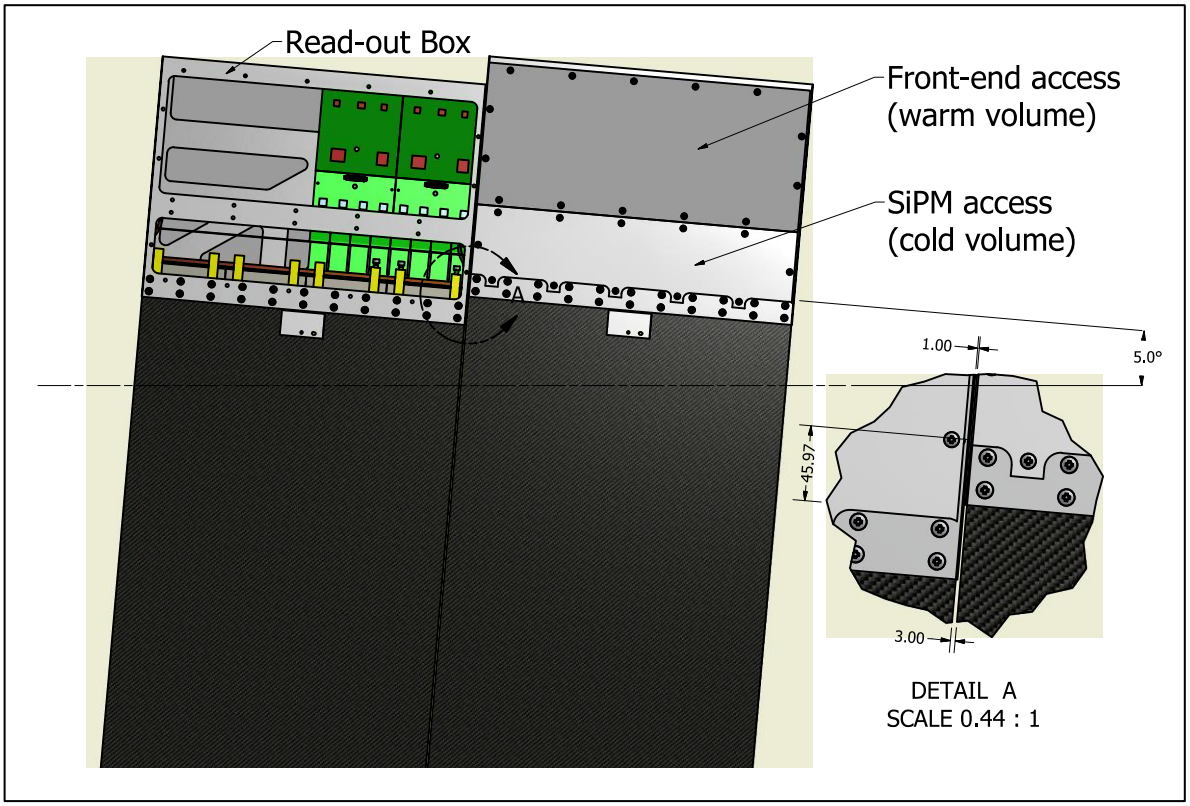


Figura 25 - Aparência final das caixas de leitura [11].

[8] Um *super-array* é formado por 4 *arrays* (128 canais cada) unidos num suporte. Quatro destes *super-arrays* (512 canais cada) são necessários para ler cada extremidade de um módulo (2.048 canais).

2.8.1. Eletrônica *Front-End* do *SciFi*

Os dispositivos da eletrônica *front-end* estão localizadas no lado quente da caixa de leitura e interagem, por um lado, com os *SiPM's* e, pelo outro, com o *back-end* do experimento. Além de viabilizar a transmissão dos dados adquiridos, estes dispositivos também distribuem tensões de polarização para os *SiPM's*, sinais de sincronismo e controle rápido (TFC, ou *Timing and Fast Control*) e comandos do sistema de controle do experimento (ECS, ou *Experiment Control System*).

A arquitetura da *front-end* do projeto *SciFi* segue diretrizes definidas no documento *Electronics architecture for the LHCb upgrade* [30]. Seu comportamento funcional é resumido pelo diagrama de blocos mostrado na Figura 26. Em cada caixa de leitura sinais analógicos provenientes de 2.048 canais *SiPM's* são amplificados, processados e digitalizados no *ASIC* **PACIFIC**. Suas saídas digitais são então enviadas ao algoritmo de clusterização rápida com o objetivo de aumentar a resolução do detector ao mesmo tempo em que reduz o volume de dados a ser transmitido. Sinais oriundos de diversos canais *SiPM* são reunidos por uma unidade lógica programável (*FPGA*) resultando na criação de *clusters*. A informação que define os clusters é serializada de acordo com as especificações do projeto GBT [31] por um *ASIC* **GBTx** dedicado e transmitida através de *links* ópticos adiante, até a sala de contagem.

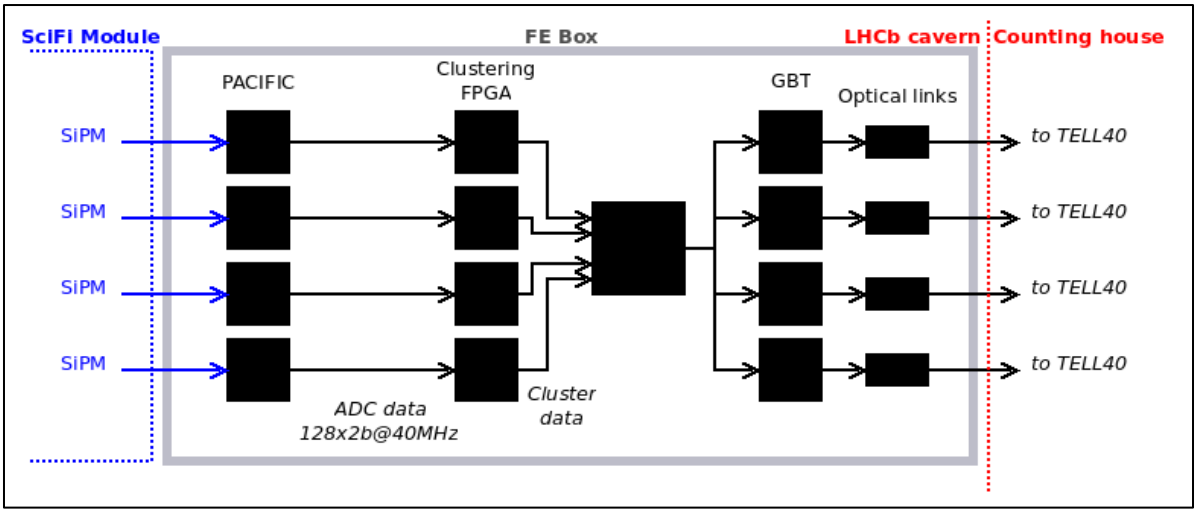


Figura 26 - Diagrama do fluxo de dados na eletrônica *front-end* [11].

No sentido contrário, *links* ópticos bidirecionais enviam comandos de controle e sinais de sincronismo da sala de contagem para os dispositivos da eletrônica *front-end*. Estes sinais são distribuídos por um *ASIC* **GBTx** dedicado. Comandos do sistema de controle do experimento são decodificados através de um *ASIC* **GBT-SCA** dedicado [32, 33]. Os chips **GBT-SCA** também fazem parte do projeto *GBT* e são capazes de encapsular sinais de controle lento (*I2C*, *JTAG*, *SPI*, *Clocks* e ADC's lentos) para transmissão de acordo com especificações do projeto *GBT*. Conversores *DC-DC* especialmente projetados para operar em ambientes com intenso campo magnético e elevada radioatividade também foram colocados na caixa de leitura para fornecer tensões de alimentação estáveis para todo o sistema de eletrônica *front-end*.

A eletrônica de *front-end* do projeto SciFi foi implementada seguindo um conceito modular, com cada caixa de leitura subdividida em dois módulos de infraestrutura (*Master Boards*), oito de clusterização (*Clusterization Boards*) e oito de aquisição analógica (*PACIFIC Boards*), com 4 *ASIC's* **PACIFIC** cada. Esta implementação está representada no diagrama da Figura 27. A montagem da Figura 28 utilizou versões-protótipo destes módulos e foi utilizada para os diversos testes.

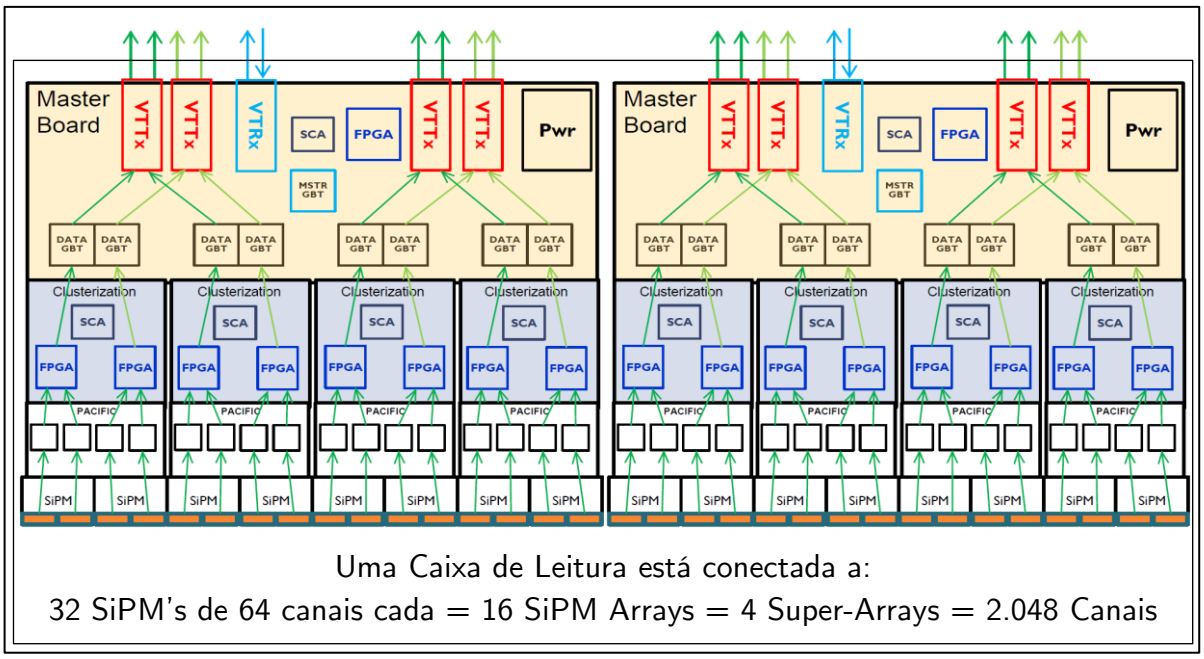


Figura 27 - Diagrama em blocos de uma caixa de leitura. Adaptado de [34].

Funcionalmente, a eletrônica de *front-end* presente numa caixa de leitura é composta de dois conjuntos formados por uma *Master Board*, quatro *Clusterization Boards* e quatro *PACIFIC Boards*. O sistema de controle do experimento considera este conjunto, esquematizado na Figura 29, como "1/2 caixa de leitura" (*Half ROB*).

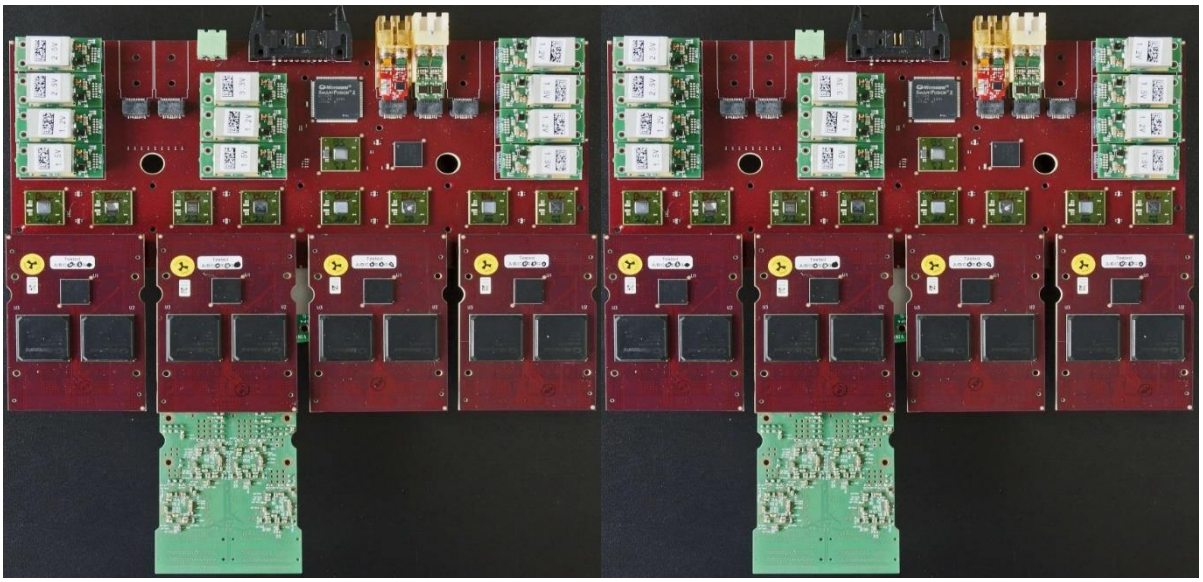


Figura 28 - Montagem (parcial) do protótipo da eletrônica *front-end*.

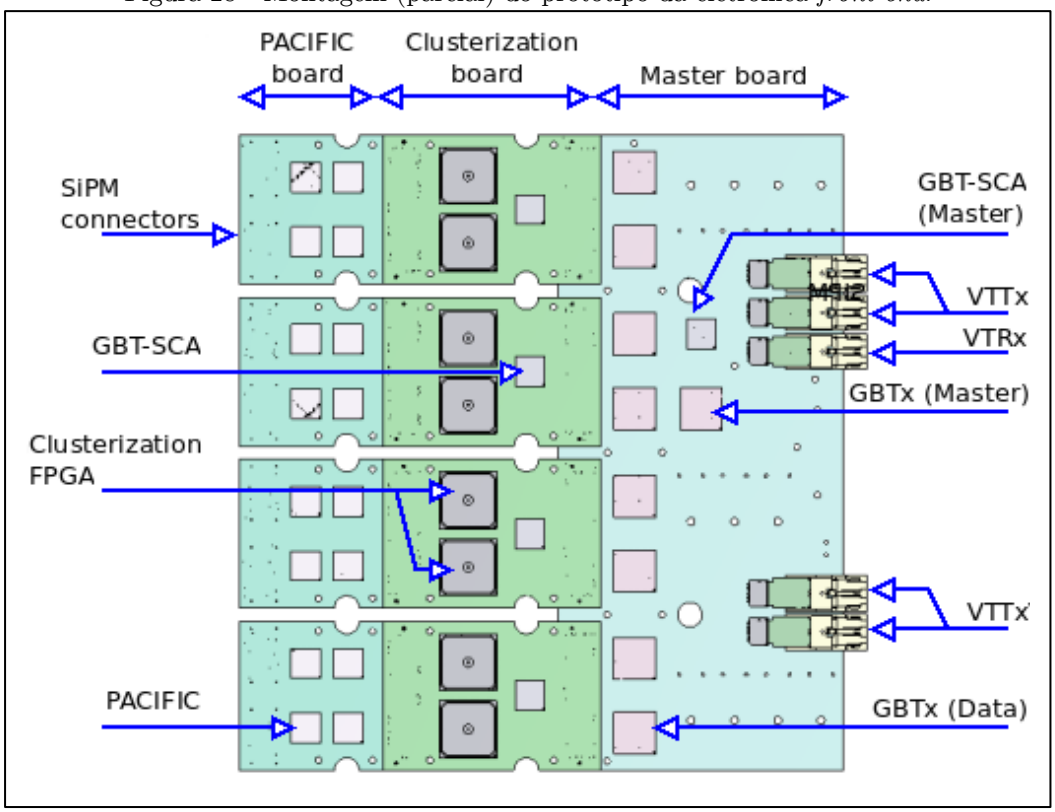


Figura 29 – Esquema da eletrônica de 1/2 caixa de leitura [35].

2.8.2. ASIC PACIFIC

O **PACIFIC** (*The low-***P***ower* **A***SIC for the s***CI***ntillating* **FI***bres tra***C***ker*) é um dispositivo microeletrônico implementado em tecnologia CMOS de $130\,\mu m$. Seu *design* é tolerante à radiação e consome menos de $10\,mW$ por canal [35]. Cada *chip* possui 64 canais de entrada que podem ser diretamente acoplados aos terminais das *SiPM's* graças ao pré-amplificador de baixa impedância ($\approx 50\,\Omega$) sensível a corrente com ajuste de nível DC presente em seu estágio de entrada.

A arquitetura de um canal de entrada do **PACIFIC** é apresentada na Figura 30. Para permitir a equalização das tensões de polarização de cada canal é possível ajustar o nível DC de cada entrada em até $700\,mV$. O ganho do estágio de entrada pode ser escolhido dentre uma das quatro opções disponíveis (0,5; 1,0; 1,5; 2,0) permitindo a operação em vários intervalos dinâmicos. O segundo estágio é um filtro configurável (*fast shaper*) que cancela a longa cauda do pulso produzido pelo *SiPM*, reduzindo a largura total a meio máximo (FWHM) para $\approx 5\,ns$. O último estágio analógico é um integrador intercalado, que alterna a cada $25\,ns$ suas duas unidades para compensar o tempo morto. O sinal integrado é amostrado pelo circuito de *Track & Hold* e digitalizado por um *ADC Flash*[9] de três limiares configuráveis. Finalmente, os canais são codificados dois a dois por quatro bits e a saída é serializada a $160\,MHz$.

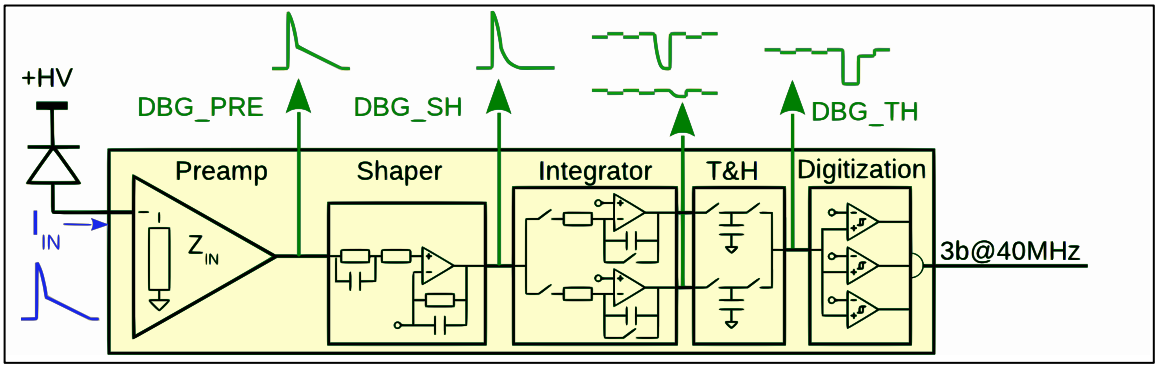


Figura 30 – Diagrama em blocos de um canal analógico no **PACIFIC**[36].

[9] Um *ADC Flash* (também conhecido como *ADC* de conversão direta) é um tipo de conversor analógico-digital que utiliza comparadores para comparar a tensão de entrada com as tensões de referência.

A seguir serão apresentados os diversos estágios do *ASIC* **PACIFIC**.

» Estágio de Entrada:

O estágio de entrada é um pré-amplificador com entrada de baixa tensão em modo de corrente, projetado para drenar a corrente de ânodo de um canal do *SiPM*. Está composto por um *current conveyor* [37] seguido por um amplificador de transimpedância, como representado pelo esquema da Figura 31. Para evitar que a forma do sinal de entrada seja alterada esta etapa possui largura de banda superior a $250\,MHz$, com impedância de entrada $\approx 50\,\Omega$ uniforme na banda passante. É possível ainda atuar no ajuste fino do nível DC do nó de entrada.

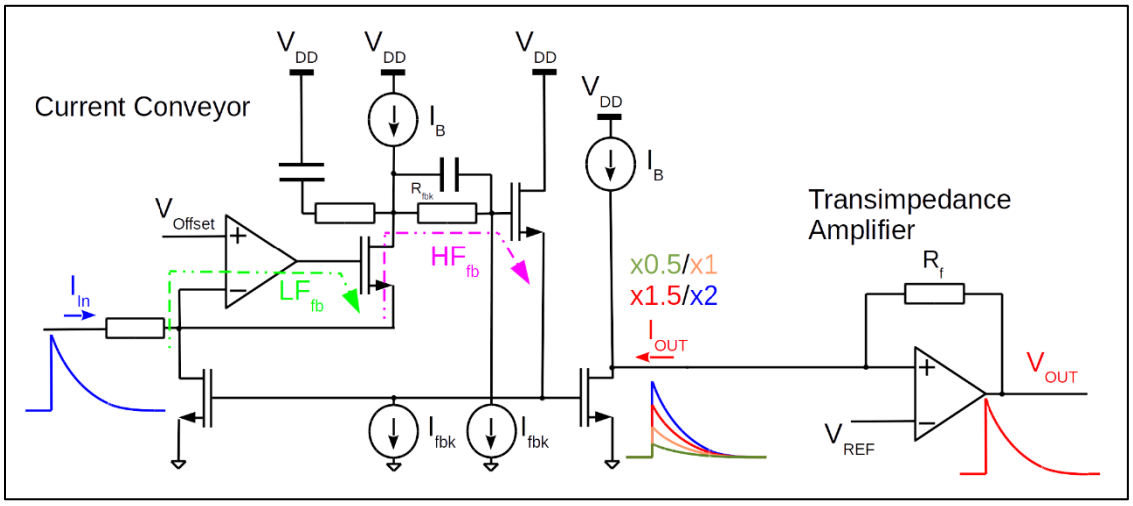


Figura 31 - Diagrama esquemático do estágio de entrada [35].

» Estágio de filtragem (*fast shaper*)

O estágio de filtragem utiliza a técnica de cancelamento de polos e zeros para reduzir a largura dos pulsos. Desta forma, a flutuação que o espalhamento temporal introduz na integração do sinal é minimizada. Este estágio foi implementado na configuração de malha fechada ilustrada na Figura 32, baseada em um amplificador operacional de transcondutância com uma rede de cancelamento em cada entrada. A rede da entrada não-inversora suprime a constante de tempo mais lenta do sinal e a rede do ramo inversor suprime a constante de tempo mais rápida. Diferentes componentes de

ambas as redes são sintonizáveis, permitindo controlar parâmetros do filtro. Por fim, um *loop* de *feedback* DC é adicionado para controlar o *offset* da saída.

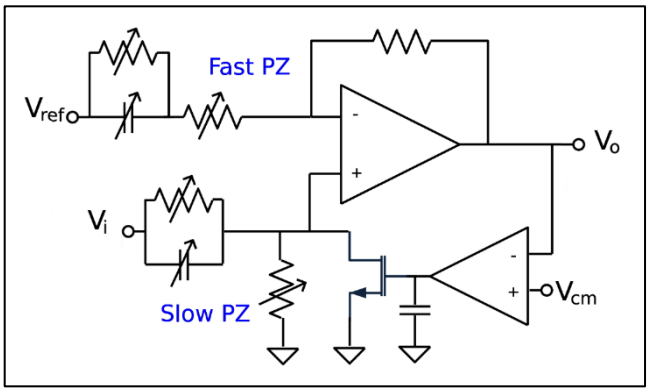


Figura 32 - Diagrama esquemático do estágio de filtragem [35].

» Estágio de integração

O estágio de integração ao final do processamento analógico acumula a carga do pulso antes de sua digitalização. É utilizada uma arquitetura de integração baseada em duas unidades complementares de amplificadores operacionais, como mostrado na Figura 33. Estas unidades são alternativamente ativadas e reiniciadas a taxa de $20\,MHz$. Desta forma uma unidade sempre está ativa enquanto a outra é reiniciada. Este *design* permite ajustar a janela de integração do **PACIFIC** ao *clock* ($40\,MHz$) do experimento, evitando qualquer tempo morto durante a aquisição.

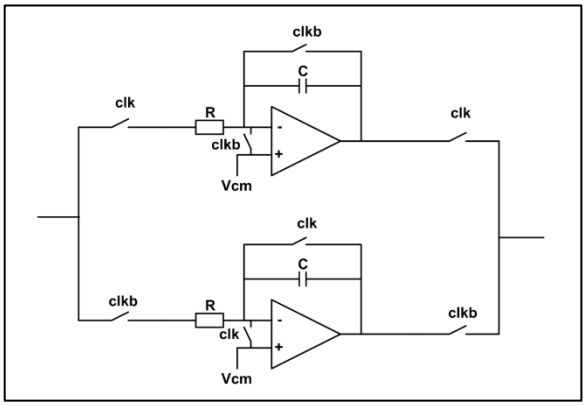


Figura 33 - Diagrama esquemático do estágio de integração [35].

O nível DC na saída de cada integrador pode ser ajustado pela tensão V_{cm} através de um conversor digital para analógico (DAC) permitindo assim compensar eventuais assimetrias entre o *offset* de unidades complementares.

» Estágio de digitalização

O estágio de digitalização foi projetado levando em conta as necessidades do processamento de clusterização. Primeiramente, o sinal é amostrado por um circuito *Track & Hold* sincronizado ao estágio de integração. Limiares configuráveis definem as transições entre os valores de um conversor analógico para digital (ADC) tipo *Flash* não-linear de três níveis. Os limiares são derivados da tensão de referência do canal e podem ser ajustados através de conversores digital para analógico ($DACs$). Cada limiar é comparado com histerese de $10\,mV$ ao nível de tensão do canal a cada $25\,ns$ por um comparador dedicado.

Maiores informações sobre este *ASIC* podem ser obtidas na referência [35]: *PACIFIC: the readout ASIC for the SciFi Tracker of the upgraded LHCb detector.*

2.8.3. Módulo de Clusterização

O módulo de clusterização (*Clusterization Board*) implementa o mecanismo discutido na Seção 2.7. O algoritmo de clusterização recebe a amplitude dos pulsos digitalizados e procura por clusters com mais de um canal disparado.

O diagrama mostrado na Figura 26 adota um FPGA para esta tarefa, oferecendo uma solução flexível para a questão da redução de dados. O procedimento de clusterização encontra-se ilustrado na Figura 34 e é explicado a seguir.

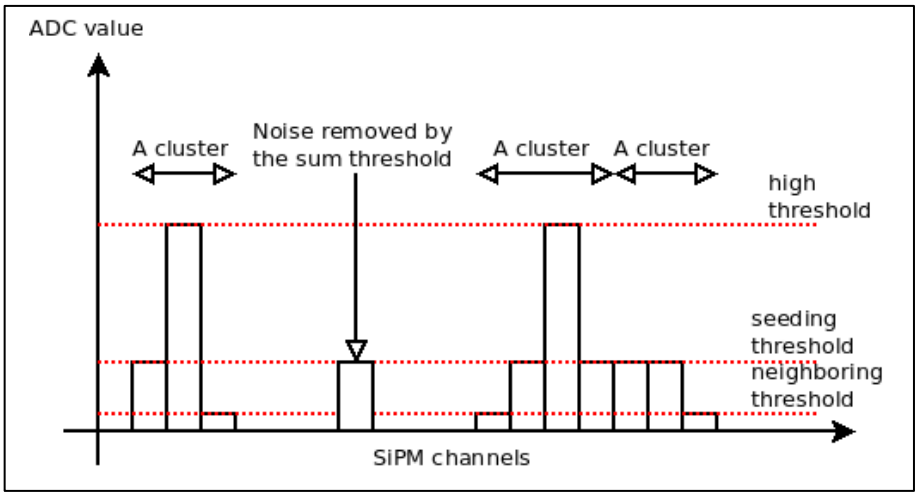


Figura 34 - Representação gráfica do algoritmo de clusterização [11].

1. *Seed threshold:* o primeiro passo no algoritmo de agrupamento é encontrar canais candidatos a formar um *cluster*, ou seja, cuja amplitude atinge o *Seed threshold*.

2. *Neighbor threshold:* canais vizinhos são incluídos num *cluster* desde que suas amplitudes atinjam o *Neighbor threshold*, o qual é inferior ao *Seed threshold*. Isso garante que o sinal máximo seja medido, pois o sinal geralmente está distribuído em mais de um canal.

3. *High threshold:* canais com amplitude acima do *High threshold* são suficientes para formar um *cluster*, mesmo sem satisfazer as condições 1 e 2.

4. *Cluster-Sum threshold:* Um *cluster* é criado apenas se a soma das amplitudes dos canais que o compõe estiver acima do *Cluster-Sum threshold*. Isso remove grande parte do ruído intrínseco ao dispositivo *SiPM*.

5. O algoritmo de clusterização admite até quatro canais em um mesmo *cluster* de modo que clusters maiores precisam ser divididos.

Clusters formados tem sua posição definida a partir de seu baricentro, calculado com precisão de 2 bits entre canais pela média ponderada abaixo:

$$baricentro = \frac{\sum_i i \times ADC(i)}{\sum_i ADC(i)} \tag{6}$$

Na Equação 6, $ADC(i)$ é um valor configurável que representa o peso atribuído ao maior *threshold* disparado no canal i. Finalmente, um pacote de tamanho fixo é gerado para cada *cluster*. Este pacote contém a posição (baricentro) do *cluster* (nº do canal *SiPM* com 7 bits mais a posição entre canais com 2 bits, resultando em uma precisão correspondente a $^1/_4$ de canal) a quantidade de canais no *cluster* (2 bits) e a soma dos valores ADC no *cluster* (3 bits).

2.8.4. Master Board

A *Master Board* é um dispositivo eletrônico de infraestrutura que interconecta todos os elementos da eletrônica de *front-end*, provê comunicação com a sala de contagem e gerencia a distribuição de energia. Suas funcionalidades são:

- Conversão de Energia (alimentação DC);
- Distribuição de TFC (*Timing and Fast Control*);
- Distribuição de comandos do ECS (*Experiment [slow] Control System*);
- Serialização e transmissão óptica de dados;

Os principais componentes embarcados na *Master Board* são um *Housekeeping FPGA*, oito chips GBT atuando como serializadores de dados, um chip GBT para controle do experimento, quatro *links* óticos unidirecionais para transmissão de dados, um bidirecional para controle do experimento e vários conversores DC-DC.

Uma descrição completa deste dispositivo pode ser consultada na referência [34], *"Master Board, technical manual"*, enquanto detalhes das interconexões entre a *Master Board* e os demais elementos da eletrônica de *front-end* são fornecidos na referência [38], *"Read Out Box interfaces"*.

2.9. MiniDAQ

O *MiniDAQ* é um dispositivo para aquisição de dados desenvolvida pelo Centro de Física de Partículas de Marselha (CPPM) com o objetivo inicial de integrar o *back-end* do experimento *LHCb*. O CBPF colaborou com este projeto, oferecendo importantes contribuições para seu sistema de controle. O *MiniDAQ*, visto na Figura 35, é composto pelos seguintes módulos:

- Um módulo *AMC* [39] (*Advanced Mezzanine Card*) composto por um *FPGA Stratix V*, 36 *links* ópticos bidirecionais e um conector *AMC* de alta densidade;

- Um módulo-base *AMC_ TP* que inclui circuitos para alimentação elétrica e distribuição de *clock* e um *CCPC* responsável pelo gerenciamento dos comandos de controle.

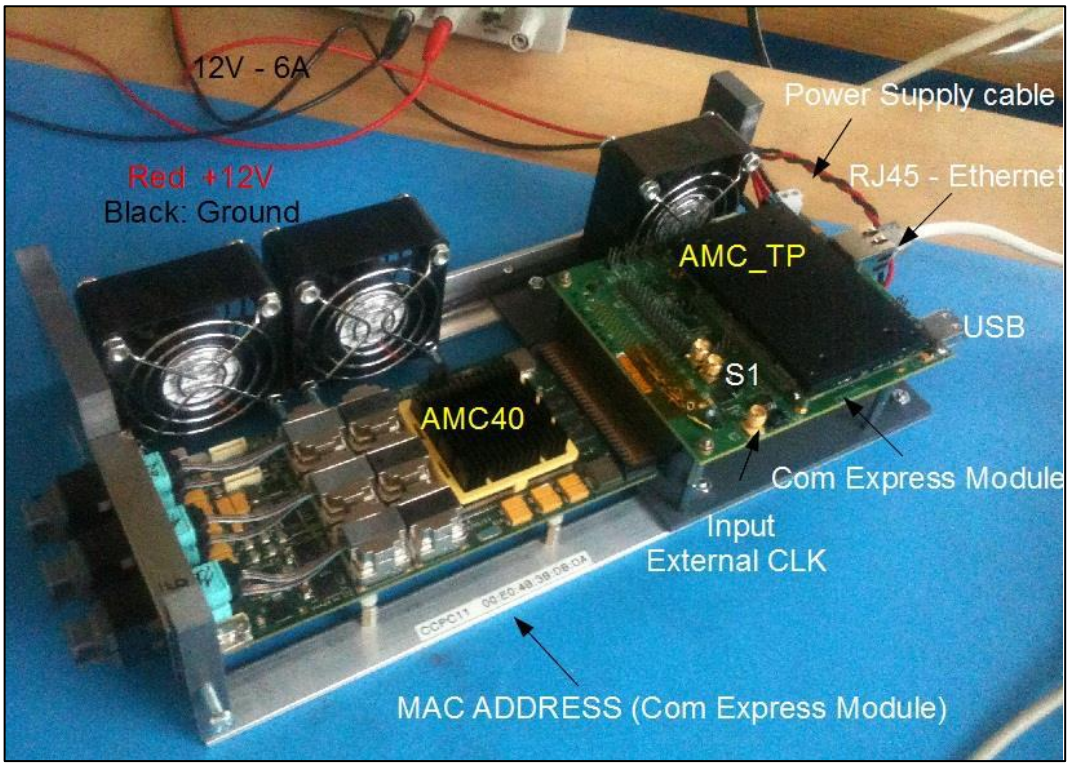


Figura 35 – Fotografia do *MiniDAQ* [40].

Vinte e quatro dos 36 *links* ópticos bidirecionais presentes no *MiniDAQ* destinam-se à conexão do módulo *AMC40* aos dispositivos eletrônicos de *front-end* através do protocolo GBT ($4{,}8\ Gbps$). Os outros 12 *links* ópticos permitem conectar o *AMC40* a um computador de aquisição através do protocolo *10-GBEth* ($10\ Gbps$). O *MiniDAQ* é então capaz de adquirir dados de uma das Caixas de Leitura apresentadas na Seção 2.8 e enviá-los para um computador através dos *links* Ethernet de $10\ Gbps$.

O controle é gerenciado pelo *CCPC*[10] modelo *Com Express KONTRON nETXe-TTc 1.0GHz 1GB/mSD*, localizado no módulo-base *AMC_TP*. Este *CCPC* executa o *driver manager*[11] e se comunica com o computador de controle através do protocolo *Ethernet* por um conector *RJ45*. Durante sua inicialização uma imagem do sistema operacional (*Linux*) deve ser transferida para *boot* pela rede. O computador de controle também é usado como um terminal para o *CCPC*.

Infelizmente, a alta demanda por transmissão de dados exigida inviabilizou sua utilização no *back-end* do experimento *LHCb* pois o *FPGA Stratix V* presente no módulo *AMC40* demonstrou não possuir elementos lógicos suficientes para sintetizar a quantidade necessária de blocos *Ethernet* e, ao mesmo tempo, alcançar a performance exigida para utilização de todos os *links* ópticos.

Ainda assim, alguns *MiniDAQ's* já produzidos são úteis para aplicações de menor escala como, por exemplo, para o sistema de teste da eletrônica de *front-end* que será apresentado no Capítulo 3, a seguir.

---

[10]CCPC, sigla em inglês para "Credit Card PC" atribuída a computadores construídos em formato compacto semelhante ao de um cartão de crédito.

[11]Driver Manager é uma parte do sistema SCADA WinCC-AO, amplamente utilizado para o controle lento, da aquisição de dados e da infraestrutura de experimentos no CERN.

Capítulo 3

Sistema de Teste da Eletrônica de *Front-End*

A eletrônica de *front-end* presente nas caixas de leitura do *SciFi*, descritas na Seção 2.8, deverão ser submetidas a uma série de testes funcionais antes da instalação de modo a garantir o perfeito funcionamento do experimento *LHCb*. A escala de produção prevista[12] e o tempo limitado para estes testes inviabilizam sua realização manual. O Dispositivo Injetor de Carga apresentado neste capítulo foi proposto como ferramenta central para viabilizar a automação dos testes da eletrônica de *front-end* das Caixas de Leitura e se apresenta como trabalho central desta dissertação, desenvolvido no âmbito deste Projeto de Mestrado.

3.1. Objetivos do Sistema de Teste

O sistema de teste proposto tem como objetivo garantir o controle de qualidade da eletrônica de *front-end*, buscando por eventuais falhas em sua fabricação e/ou montagem, e deve ser capaz de:

- Identificar canais defeituosos (*dead channels*) e ruidosos (*noisy channels*), bem como interferências (*crosstalk*) e diferenças de fase (*timing*) entre canais;
- Testar o funcionamento do *ASIC* **PACIFIC**, verificando a uniformidade dos níveis DC e ganhos de entrada e das tensões de *threshold*, o funcionamento dos *shapers*, dos integradores, dos comparadores e da lógica digital;

[12] O layout das estações SciFi (ver Seção 2.4) prevê 144 módulos detectores, cada um com duas caixas de leitura, uma em cada extremidade. Serão produzidas, portanto, 288 caixas de leitura para completar a montagem e mais algumas ($\approx 10\%$) de reserva, totalizando mais de 300 caixas.

- Verificar a integridade de uma medição física emulada através da injeção de cargas equivalentes àquelas geradas pelas fotomultiplicadoras quando excitadas por eventos reais. Esta verificação deve ser capaz de validar o funcionamento de toda a cadeia de aquisição, incluindo os mecanismos de clusterização, de transmissão e de armazenagem dos dados.

3.2. Conceituação do Sistema de Teste

As seguintes opções foram inicialmente consideradas para o estabelecimento do conceito de construção do sistema de teste proposto:

- *SiPM* multiplexada:

Inicialmente cogitou-se utilizar um LED para excitar uma fotomultiplicadora *SiPM* real cuja saída elétrica estaria conectada a um multiplexador de 2048 saídas, como visto na Figura 36. Este conceito demonstrou-se inexequível devido à complexidade e dificuldades de ordem prática para construção de um multiplexador analógico com 2048 saídas capaz de garantir boa relação sinal-ruído com a baixa potência oriunda da fotomultiplicadora. Esta opção foi prontamente descartada.

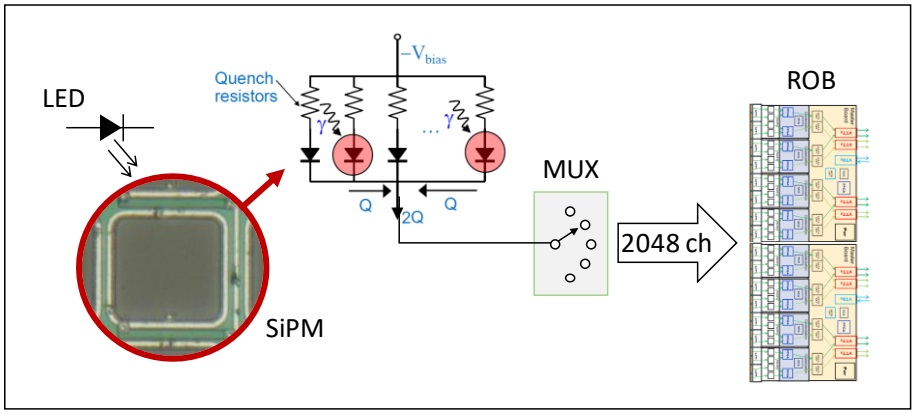


Figura 36 – Diagrama do conceito "*SiPM* multiplexada".

- Gerador de sinais com *fan-out* comutado por relé:

Cogitou-se ainda utilizar um gerador de funções de bancada alimentando um amplificador *fan-out* cujas saídas seriam conectadas a relés capazes de comutar individualmente, como visto na Figura 37, os 2048 canais de saída. Este conceito foi implementado no sistema de teste das estações T que estão atualmente em operação no *LHCb* sendo, assim, considerado exequível. Entretanto, a inflexibilidade do dispositivo acarretaria no cumprimento apenas parcial dos objetivos definidos na Seção 3.1. Por isso, optou-se por considerar uma solução mais completa.

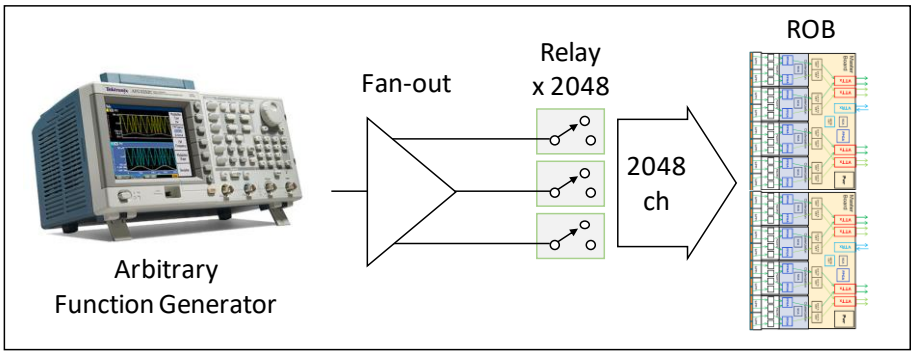


Figura 37 – Diagrama do conceito "Gerador com *fan-out* comutado por relé".

- Dispositivo Injetor de Carga:

Por fim, considerou-se utilizar *FPGA's* para a produção de pulsos com amplitude binária, mas arbitrários no domínio do tempo, e utilizar circuitos analógicos para modular sua amplitude, como visto na Figura 38. Esta opção se mostrou exequível e adequada ao cumprimento dos objetivos definidos em 3.1, tendo sido escolhida para a implementação do presente sistema de teste.

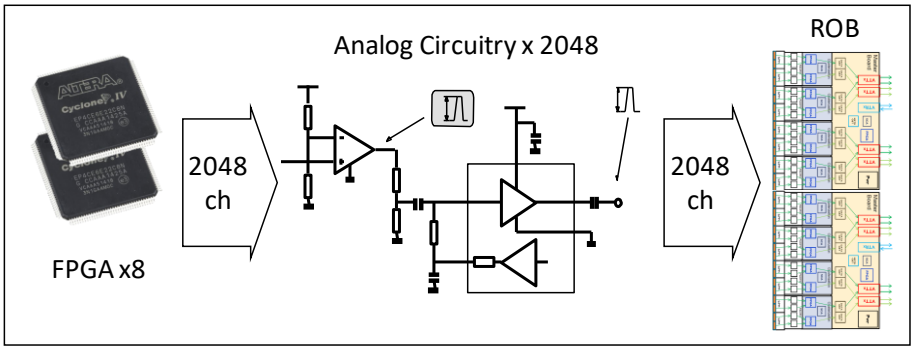


Figura 38 – Diagrama do conceito "Dispositivo Injetor de Carga".

3.3. Arquitetura do Sistema de Teste

A arquitetura do sistema de testes foi pautada em compatibilidade e interoperabilidade com a arquitetura já desenvolvida (ver Seção 2.8.1) para o controle e a aquisição de dados na *front-end* do projeto *SciFi*. A compatibilidade permite o aproveitamento de dispositivos de *hardware* já desenvolvidos para o projeto *SciFi*, e a interoperabilidade permite que todos os componentes do sistema de testes estejam integrados em uma plataforma única.

O dispositivo *MiniDAQ* foi escolhido por atuar como elemento centralizador, capaz de concentrar em um único nó os controles lento e rápido e a temporização do sistema. A implementação de algoritmos de teste fica facilitada à medida que se torna possível o uso do Sistema Supervisório *WinCC* para execução de códigos e criação de painéis para interface com o usuário. A arquitetura proposta encontra-se apresentada no diagrama em blocos da Figura 39.

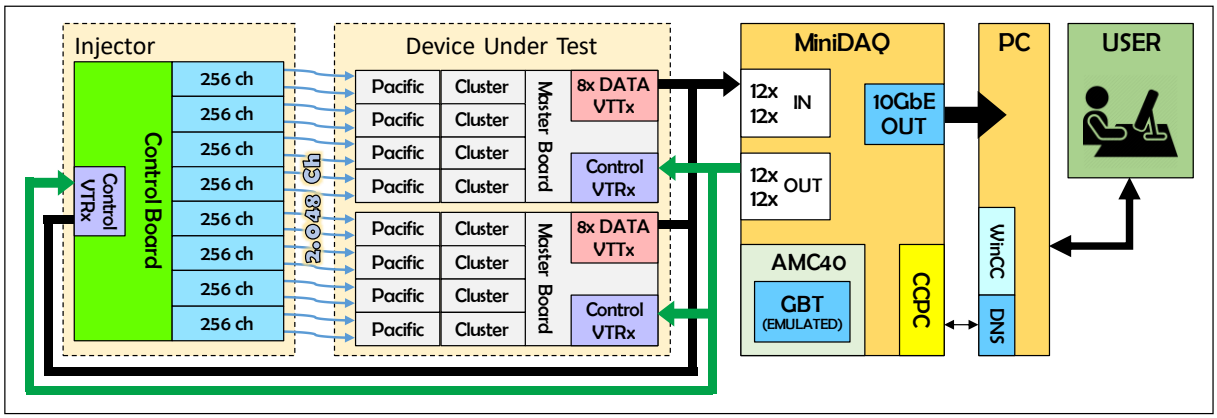


Figura 39 – Diagrama em blocos do sistema de testes.

3.4. Dispositivo Injetor de Carga

O Injetor de Carga é um Dispositivo capaz de emular os sinais elétricos produzidos por quatro *super-arrays SiPM* que estejam sendo excitados por fótons produzidos por fibras cintilantes sob efeito da radiação ionizante. Esta emulação tem como objetivo excitar de maneira controlada o mecanismo de leitura apresentado na Seção 2.7.

A emulação dos canais *SiPM* ocorre através da substituição dos elementos fotomultiplicadores por canais de saída do Dispositivo Injetor de Carga, capazes de injetar nas entradas do *ASIC* **PACIFIC** uma quantidade de carga equivalente àquela produzida pela *SiPM*. Para a avaliação completa de uma Caixa de Leitura são necessários 2.048 canais de injeção, cada um atuando como um gerador independente de pulsos.

A excitação do mecanismo de leitura (Seção 2.7) no intervalo dinâmico adequado requer que o intervalo de variação da carga elétrica dos pulsos injetados seja equivalente àquele que é esperado para a quantidade de pixels (ver Figura 20, na página 28) disparados por detecção, isto é, entre 1 e 5 fotoelétrons por canal por evento. A quantidade de carga necessária pode então ser estimada[13]:

$$Q_{fotoelétron} = Ganho_{fotomultiplicadora} \cdot Q_{elétron} \tag{7}$$

$$Q_{fotoelétron} = 2{,}7 \times 10^{6} \cdot 1{,}6 \times 10^{-19} = 0{,}43\ pC$$

$$Q_{pulso} = Q_{fotoelétron} \cdot n \quad | \quad 1 \leq n \leq 5 \tag{8}$$

$$0{,}43\ pC \ \leq Q_{pulso} \leq 2{,}15\ pC$$

[13]A estimativa da quantidade de carga depende do ganho esperado para a fotomultiplicadora, definido como o número de elétrons provocados pelo disparo de um pixel por um único fóton. Apesar deste ser um parâmetro crucial para um *SiPM*, seu valor é difícil de estimar pois depende da quantidade de carga acumulada na região de depleção do semicondutor. Esta quantidade de carga, por sua vez, possui elevada incerteza pois está relacionada a fatores de construção, tensão de polarização e temperatura operacional de cada canal do *SiPM*. Na equação 7 foi considerado o ganho de fotomultiplicação médio observado sob condições normais de operação nas *SiPM's* já fabricadas para o projeto *SciFi*.

A quantidade de carga por pulso injetado pode ser controlada através da manipulação da largura, amplitude e/ou forma do pulso. Entretanto, o pulso resultante deve ser largo o suficiente para se propagar pelo circuito e curto o suficiente para ocupar uma janela de integração ($25\,ns$) do *ASIC* **PACIFIC**. Sua amplitude deve respeitar o limite dinâmico dos circuitos de processamento analógico do *ASIC*.

Optou-se pela fragmentação do Dispositivo Injetor de Carga em diversos Módulos Injetores, cada um com uma fração dos canais necessários. Desta forma, cada Módulo Injetor (*Injector Board*) foi projetado como uma unidade autônoma geradora de sinais com 256 saídas independentes. Vários Módulos Injetores podem, entretanto, se associar para completar a quantidade demandada de canais. Um Módulo de Controle (*Control Board*) foi projetado para distribuir sinais de controle e temporização a diversos Módulos Injetores, para que estes possam trabalhar em conjunto. O *MiniDAQ* centraliza comandos de controle e assegura a base de tempo homogênea. A Figura 40 ilustra esta subdivisão.

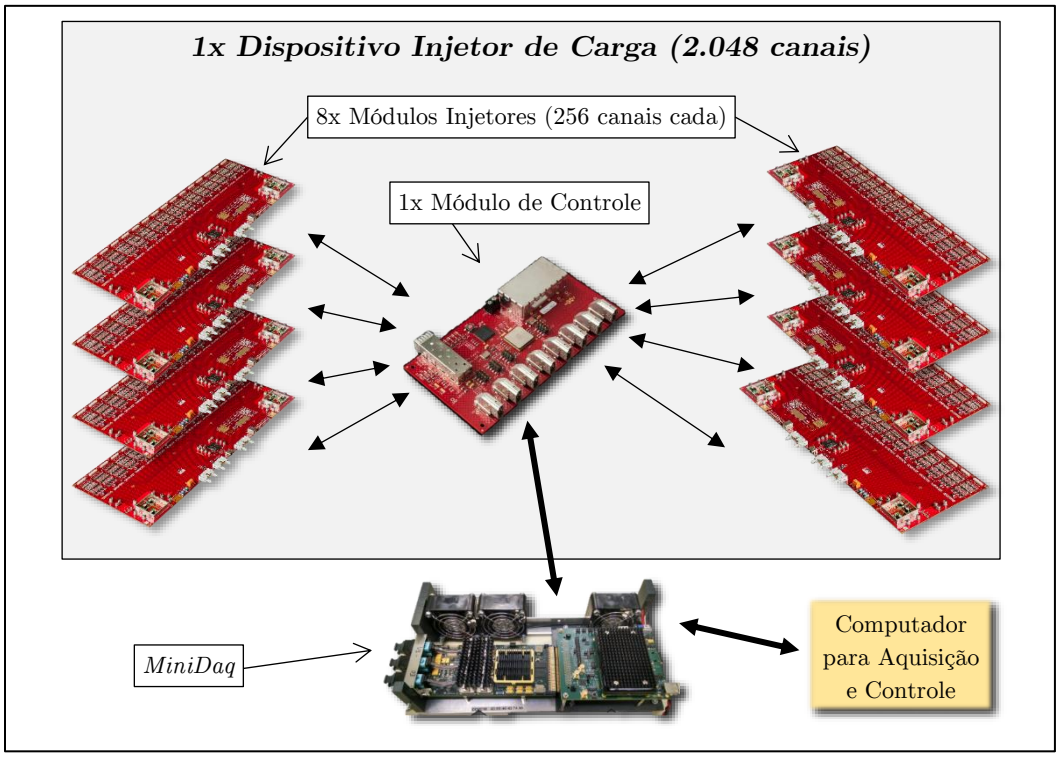


Figura 40 – Dispositivo Injetor de Carga.

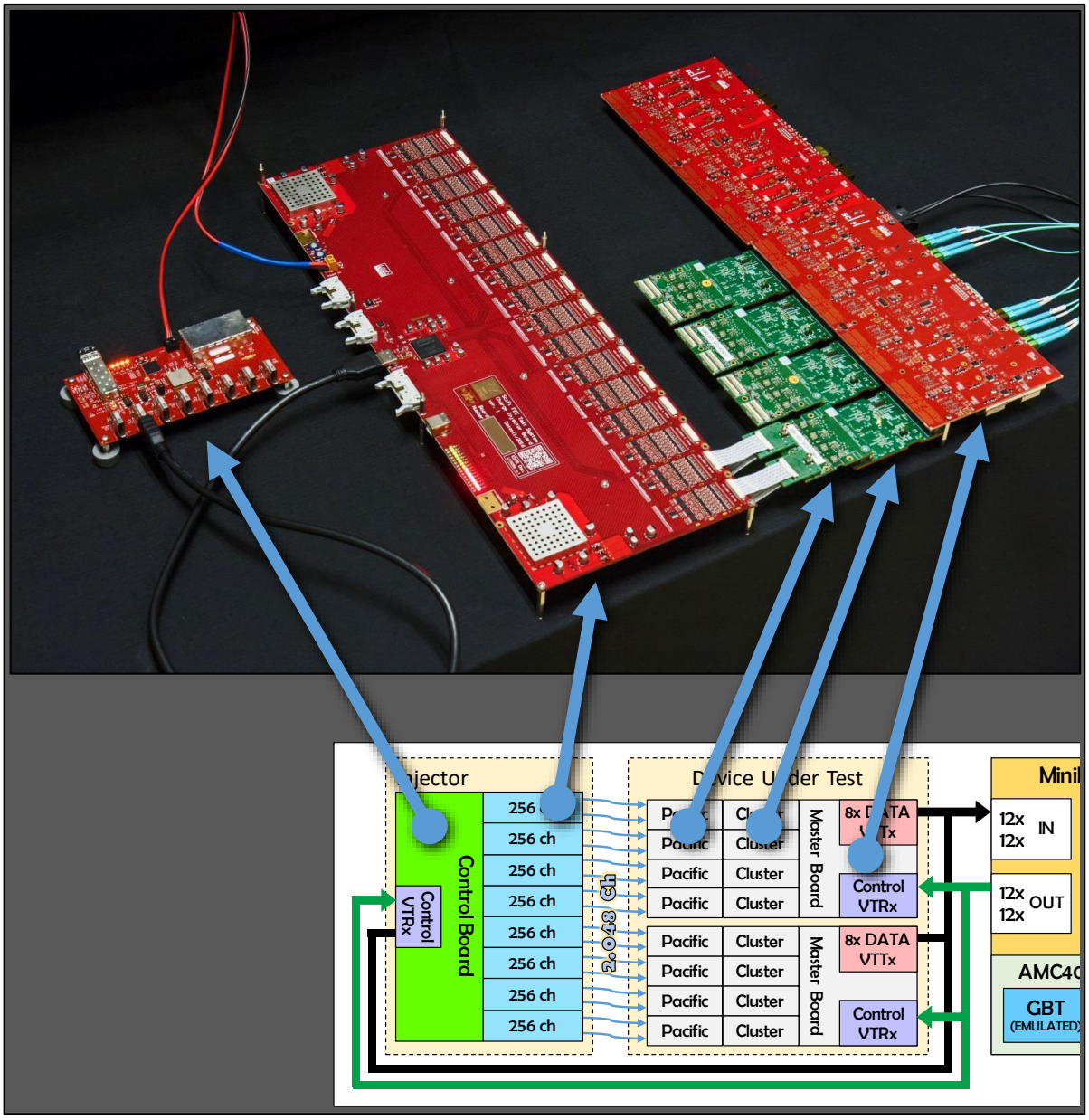


Figura 41 – Montagem parcial do Dispositivo Injetor de Carga acoplado a uma Caixa de Leitura. As setas indicam a correlação da montagem com o diagrama em blocos da Figura 39, reproduzido parcialmente abaixo.

3.5. O Módulo Injetor

O Módulo Injetor é um gerador de sinais multicanal com 256 saídas. Este módulo foi projetado como uma unidade autônoma, dotada de mecanismos que permitem sua operação em conjunto com outros Módulos Injetores para ampliar a quantidade de canais disponíveis.

Seu projeto norteou-se pelo cumprimento dos objetivos estabelecidos na Seção 3.1 através da arquitetura apresentada na Figura 39 – Diagrama em blocos do sistema de testes. Para isto, foram concebidos blocos de circuitos digitais – majoritariamente implementados através de *FIRMWARE* em um *FPGA Altera Cyclone IV* – responsáveis por controlar e temporizar o módulo e por disparar os pulsos, e circuitos analógicos com característica predominantemente multicanal que atuam na modulação de amplitude dos pulsos e no ajuste da impedância de saída. Os circuitos eletrônicos que compõem este módulo são apresentados no diagrama em blocos da Figura 42 e, mais detalhadamente, nos diagramas do Apêndice B.

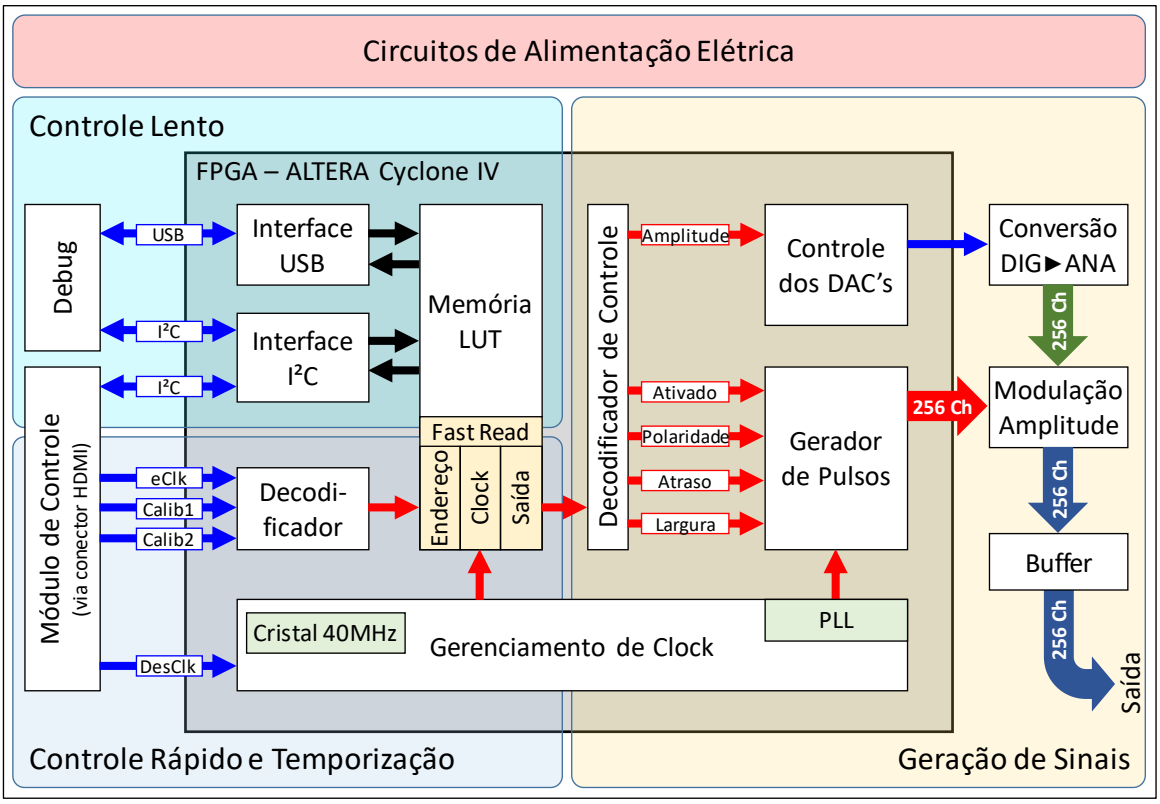


Figura 42 – Diagrama em blocos do Módulo Injetor.

3.5.1. Circuitos Digitais

Os circuitos listados nesta Seção foram implementados em um dispositivo reprogramável embarcado do tipo *FPGA* (*Field Programmable Gate Array*) modelo *EP4CE30F23* (família *Cyclone IV*) da fabricante *Altera Corporation*. O uso de um dispositivo reprogramável permite aprimoramento contínuo do projeto visto que o *FPGA* pode ser facilmente reprogramada através dos conectores *JTAG* e *Active Serial* presentes no Módulo Injetor. As conexões elétricas do *FPGA* encontram-se descritas do Diagrama 8 ao Diagrama 17 do Apêndice B. Os circuitos a seguir foram implementados através do *firmware* transcrito no Apêndice D.

» Gerenciamento de *clock*:

É um bloco *PLL* gerado automaticamente pelo *software Quartus II*. Possui uma entrada para *clock* externo e outra para o cristal de *clock* interno, permitindo que o Módulo Injetor opere tanto em modo de sincronismo externo quanto em modo autônomo, produzindo sua própria base de tempo. Quando múltiplos módulos são requeridos (como no Dispositivo Injetor) a utilização de uma fonte de sincronismo externa torna-se imperativa devido a necessidade de uma base de tempo única e compartilhada por todos os módulos. O circuito *PLL* produz ainda um *clock* secundário de $400\,MHz$ para o bloco gerador de pulsos.

» Memória *LUT* e suas interfaces:

Memória rápida do tipo *LUT* (*Lookup Table*) que armazena o banco de registradores responsáveis pelo controle lento[14] do injetor. Diversas interfaces permitem acesso para leitura e escrita deste banco de registradores. Estas interfaces podem operar de forma simultânea, facilitando eventuais depurações do sistema.

---

[14] O controle lento do Módulo Injetor será apropriadamente descrito na Seção 3.8

» Decodificador de Endereços:

Recebe e decodifica duas linhas de controle (*Calib1* e *Calib2*) de alta velocidade[15] provenientes do Módulo de Controle. A memória LUT pode, desta maneira, ser endereçada para leitura com 4 bits à plena taxa ($40MHz$) permitindo selecionar a cada ciclo de *clock* um dos 16 possíveis cenários de injeção armazenados no banco de registradores. Cada cenário contém todas as configurações para todos os canais do injetor.

» Decodificador de Controle:

Decodifica a região do banco de registradores correspondente ao cenário atualmente selecionado pelo decodificador de endereços fornecendo os comandos necessários para controlar os $DAC's$ e o bloco gerador de pulsos.

» Controle dos $DAC's$:

O circuito de controle dos $DAC's$ gerencia as linhas seriais que controlam os conversores digital-analógico a partir de parâmetros do banco de registradores. Foram incluídas quatro linhas seriais para obter uma maior taxa de atualização dos conversores através da distribuição da largura de banda, como visto na Figura 43. Quatro deserializadores reduzem a quantidade de pinos do $FPGA$ para as linhas de seleção (*Chip Select*). Devido ao elevado comprimento das trilhas e à grande quantidade de cargas em paralelo alguns *drivers* foram empregados ao longo das linhas.

[15] As linhas *Calib1* e *Calib2* operam a $40\ MHz$ em DDR (*Double Data Rate*), totalizando $160\ Mbps$.

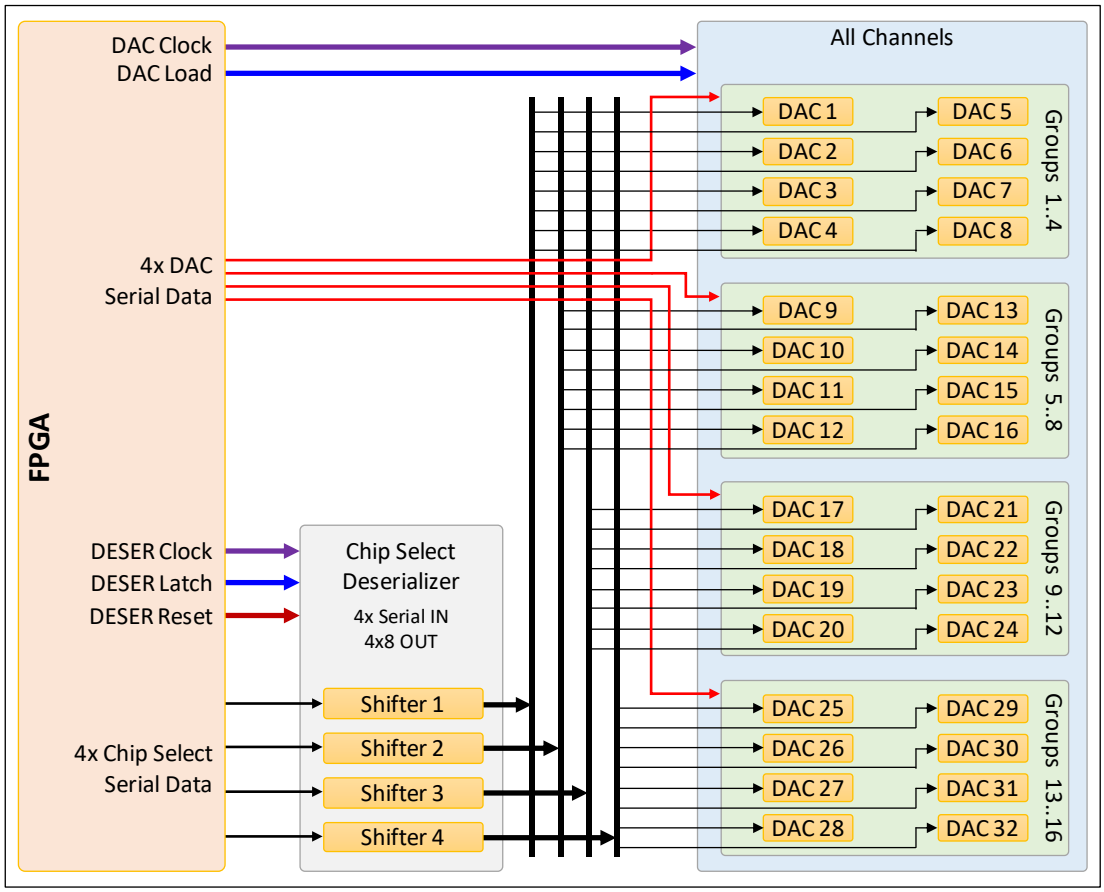


Figura 43 – Esquema de controle dos *DAC's*.

Como de-serializador foi utilizado o circuito integrado *74LVC594A* da fabricante *NXP Semiconductors*. Como *driver*, foi utilizado o circuito integrado comparador de alta velocidade modelo *ADCMP600BRJZ* da fabricante *Analog Devices*. Estas implementações são externas ao FPGA e encontram-se detalhadas no Diagrama 18 e no Diagrama 19, ambos no Apêndice B.

» Gerador de Pulsos:

Produz pulsos digitais de amplitude fixa e largura variável. É composto por circuitos temporizadores que controlam as saídas de pulso a partir das bases de tempo fornecidas ($40MHz$ e $400MHz$) e dos parâmetros de controle decodificados do banco de registradores. Cada canal tem seu estado de ativação (ligado/desligado) e polaridade manipulados individualmente e as configurações de atraso e de largura são globais.

3.5.2. Circuitos Analógicos

Os blocos listados a seguir são de natureza predominantemente analógica e foram implementados na forma de circuitos eletrônicos montados sobre a placa de circuito impresso do Módulo Injetor. Uma visão geral destes blocos é fornecida pelo Diagrama 4 do Apêndice B.

» Conversores de digital para analógico

A conversão digital-analógico para controle da amplitude do pulso tem como base o circuito integrado *MAX5259EEE+* da fabricante estadunidense *Maxim Integrated Products*. Foram utilizados 32 circuitos deste tipo e cada possui oito canais de conversão, totalizando os 256 canais demandados pelo Módulo Injetor. Os circuitos integrados foram distribuídos em ambas as faces da placa de circuito impresso na forma de pares alinhados ao longo de sua maior dimensão. Cada par está próximo ao grupo de modulação de amplitude ao qual suas saídas encontram-se conectadas visando uma redução de interferências e a otimização do roteamento de trilhas. O circuito elétrico deste bloco é apresentado na Figura 44.

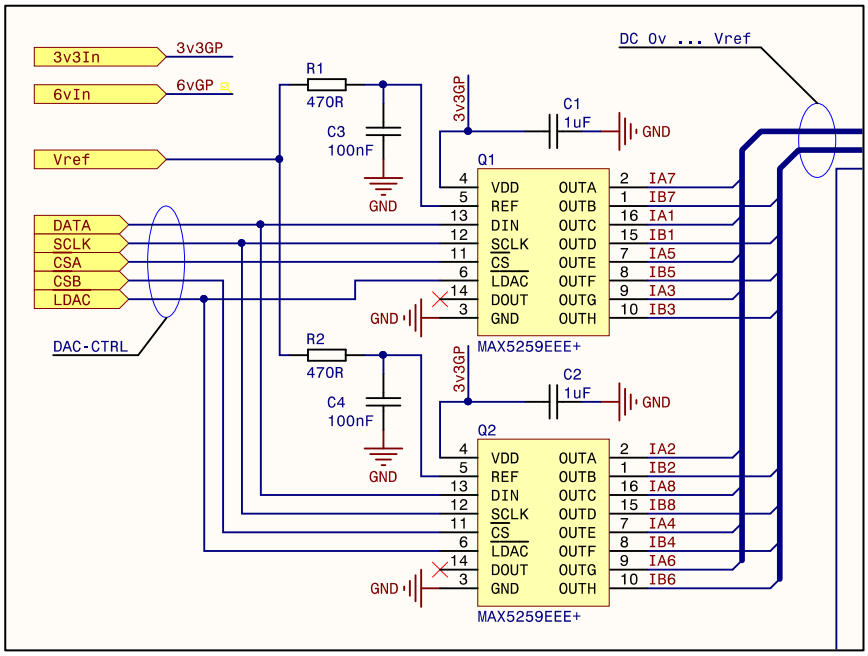


Figura 44 – Conversores digital-analógico. Extraído do Diagrama 5 do Apêndice B.

» Modulação de Amplitude:

A amplitude dos pulsos injetados é controlada modulando as amplitudes dos pulsos digitais produzidos pelo bloco gerador de pulsos. Para tal foram utilizados comparadores de alta velocidade operando como se fossem *Level Shifters*, isto é, com o pino de alimentação (V_{cc}) conectado a uma saída de um dos conversores digital-analógico. Levando em conta a incapacidade de os conversores fornecerem a corrente elétrica necessária para o funcionamento dos comparadores e a necessidade de adequar as tensões de alimentação ao intervalo permitido pelo fabricante ($2.5\,V\ < V_{cc} <\ 5.5\,V$) foram utilizados arranjos de amplificadores operacionais entre as saídas dos conversores digital-analógico e os pinos de alimentação dos comparadores. As saídas dos comparadores estão conectadas a divisores resistivos simples que atenuam por um fator de $2{,}54 \times 10^{-3}$ a amplitude dos pulsos de modo a adequar a quantidade de carga injetada ao intervalo definido pela Equação 8. Neste bloco foram utilizados dois modelos de circuitos integrados, o comparador *ADCMP600BRJZ* da fabricante estadunidense *Analog Devices* e o amplificador operacional quádruplo de alta performance *MCP6L04T-EST* fabricado pela também estadunidense *Microchip Technology Inc.* O circuito elétrico deste bloco é apresentado na Figura 45.

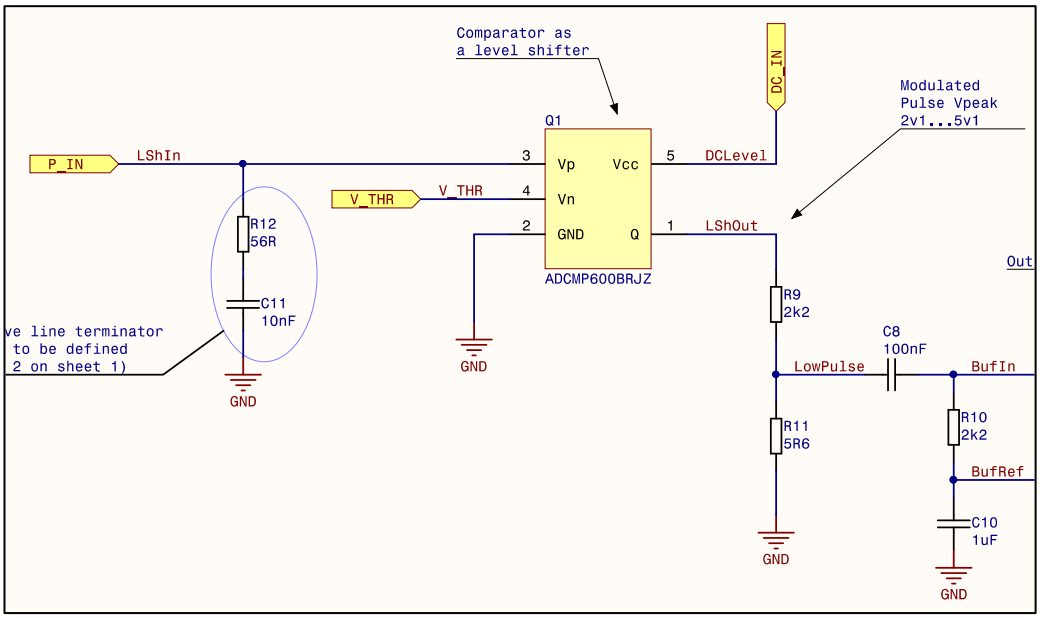


Figura 45 – Modulador de Amplitude. Extraído do Diagrama 7 do Apêndice B.

» *Buffer* de Saída

Este estágio final tem como objetivo propiciar o acoplamento apropriado das saídas do Módulo Injetor com as entradas de baixa impedância ($\approx 50\Omega$) do *ASIC* **PACIFIC**. Isto é necessário pois o divisor resistivo ao final do estágio anterior impede seu acoplamento direto com a linha de transmissão por conta do indevido casamento das impedâncias. O *buffer* de *loop* fechado de alta velocidade ($Slew\ Rate \approx 8000V/\mu s$) modelo *BUF602IDBVT* da fabricante *Texas Instruments Inc.* foi utilizado para a implementação deste estágio. Seu circuito elétrico é apresentado na Figura 46.

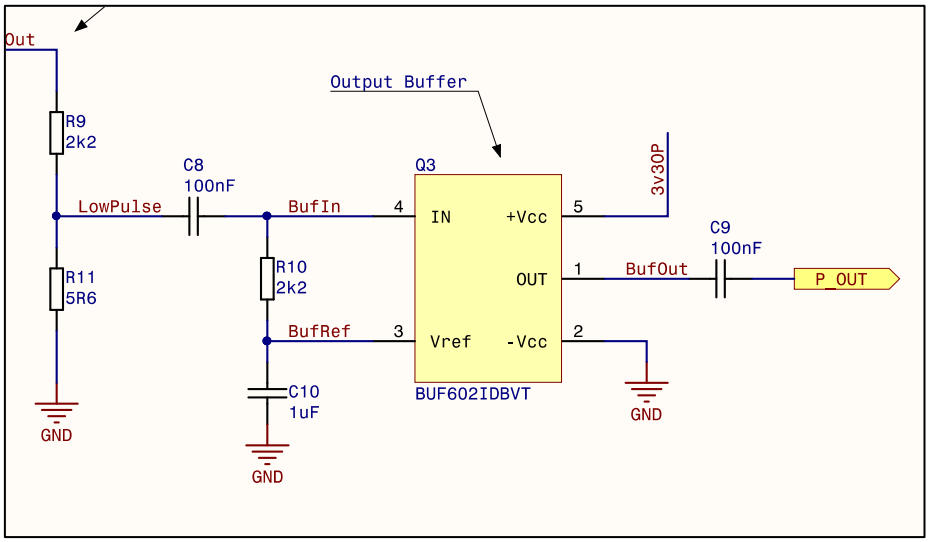


Figura 46 – *Buffer* de Saída. Extraído do Diagrama 7 do Apêndice B

» Alimentação Elétrica

Os circuitos de alimentação elétrica regulam os diversos níveis de tensão ($1{,}2\,V;\ 2{,}5\,V;\ 3{,}3\,V;\ 6{,}0V$) necessários ao funcionamento do Módulo Injetor. Estes circuitos estão divididos nos extremos da PCI de modo a minimizar quedas de tensão ao longo dos planos de alimentação. Linhas de alimentação específicas para circuitos digitais foram criadas de modo a reduzir interferências de chaveamento. A entrada de força recebe uma linha de $12\,V_{dc}\ /\ 4\,A$ proveniente de um adaptador AC externo. Diversos reguladores DC-DC de alta performance *PTH08080WAH* e *PTN78000WAH*, ambos da fabricante *Texas Instruments Inc.*, foram empregados.

Os circuitos reguladores das tensões $6v$ e $3v3$ são apresentados na Figura 47 e os reguladores das tensões $1v2$ e $2v5$ na Figura 48.

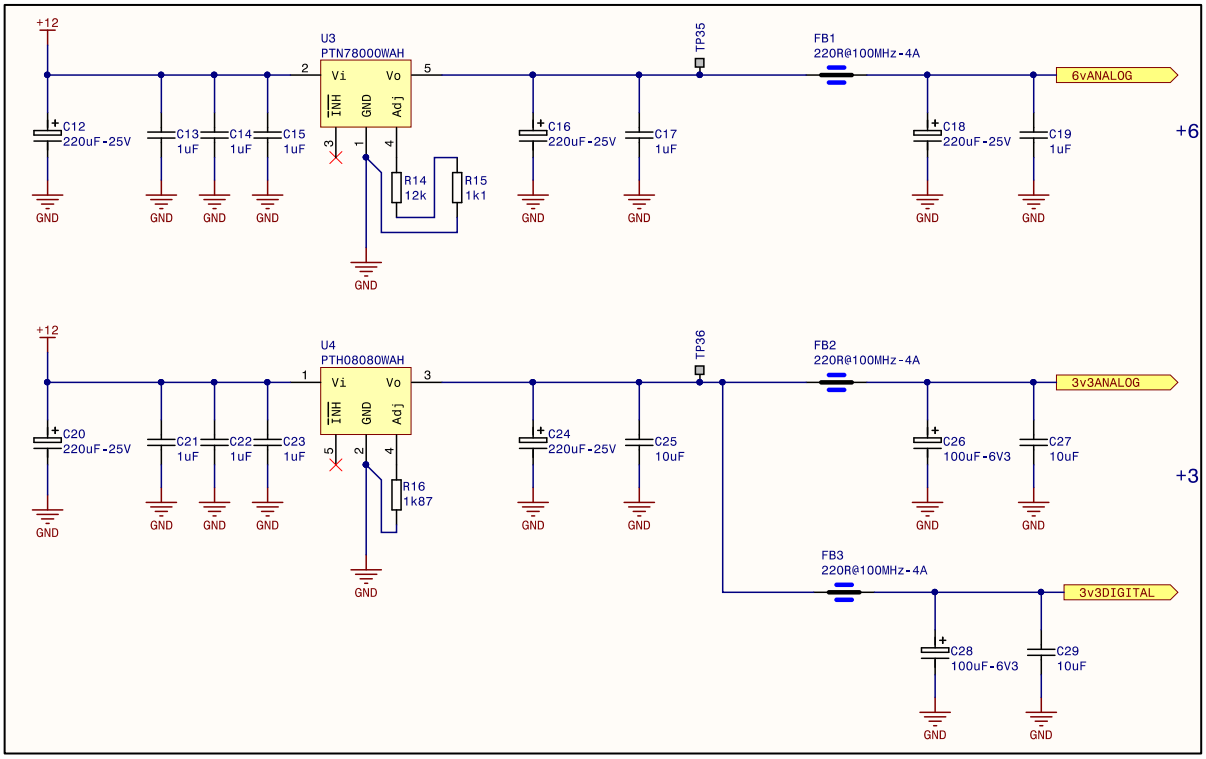


Figura 47 – Reguladores de tensão $6v$ e $3v3$. Extraído do Diagrama 2 do Apêndice B.

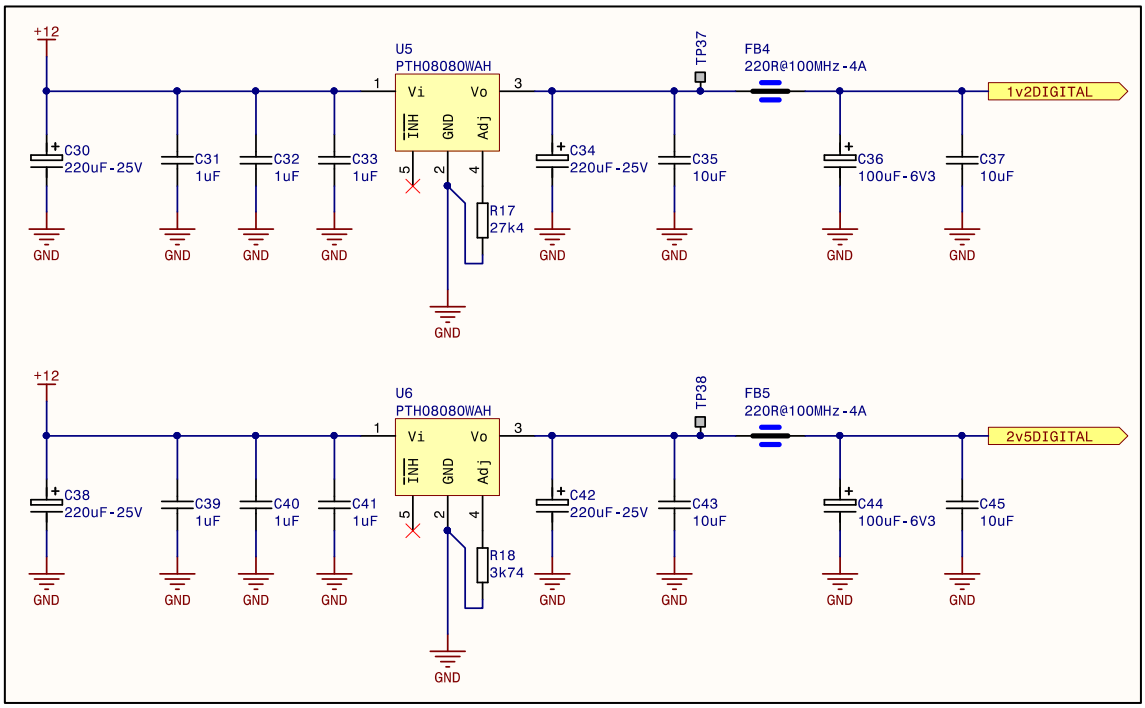


Figura 48 – Reguladores de tensão $1v2$ e $2v5$. Extraído do Diagrama 3 do Apêndice B.

3.5.3. Cabeamento das Saídas de Pulso

A implementação do *layout* da Figura 40 requer a conexão das 2.048 saídas do Dispositivo Injetor de Carga às 2.048 entradas da Caixa de Leitura. Esta conexão não deve interferir no alinhamento temporal entre os canais, devendo possuir linhas de transmissão eletromagneticamente idênticas para todos os canais. O uso de cabeamento convencional mostrou-se demasiado complexo devido à geometria requerida para a montagem. Optou-se então pela fabricação do multicabo na forma de um circuito impresso multicamada flexível com a geometria apresentada na Figura 49. Sua produção na forma de circuito impresso permite que o desenho das trilhas possua comprimentos idênticos, mesmo com diferentes comprimentos dos cabos.

Devem ser produzidos, pelo menos, 16 multicabos para cada Dispositivo Injetor de Carga construído. Unidades adicionais também devem ser produzidas visto que a vida útil do conector[16] utilizado nas entradas da Caixa de Leitura é limitada e pode levar ao descarte antecipado dos multicabos.

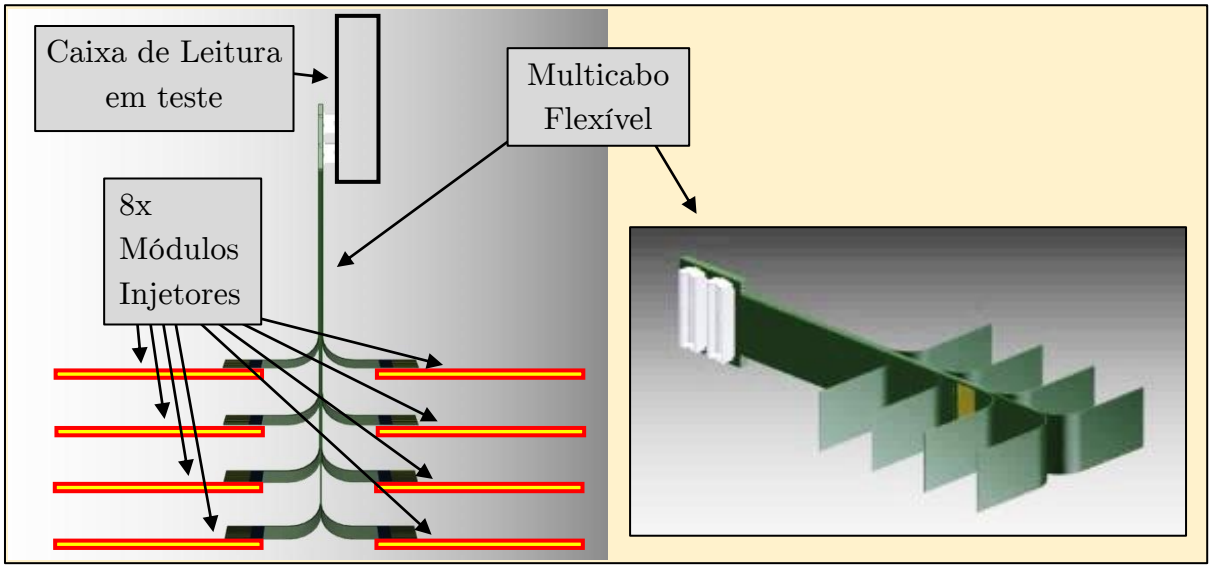


Figura 49 – Esquema de cabeamento do Dispositivo Injetor de Carga.

[16]Conector FPC Hirose DF12(3.0)-80DP-0.5V: Especificado para, no máximo, 50 ciclos de inserção. Deve-se também evitar, na medida do possível, manipular as saídas do Módulo Injetor pois o conector FPC Molex 54104-3331 utilizado está especificado para, no máximo, 20 ciclos de inserção.

3.6. Desenvolvimento do Protótipo de 8 Canais

Definidos os principais conceitos do projeto, e com o objetivo de colocá-los à prova, deu-se início à produção do protótipo de 8 canais[17] visto na Figura 50. Sua montagem teve como base uma PCI fabricada no Brasil pela Micropress Circuitos Impressos com especificações e desenhos produzidos com auxílio da ferramenta *Altium Designer*.

Tendo em vista que o protótipo deveria ser montado manualmente levou-se em conta uma forma de embarcar o *FPGA* sem necessidade de ter que soldá-lo na PCI pois seu encapsulamento *BGA* obriga a utilização de métodos mecanizados de montagem que não haviam sido previstos. O cartão *CoreEP4CE10* fabricado pela chinesa *Waveshare* demonstrou ser uma solução bastante conveniente para resolver este problema por integrar um *FPGA EP4CE10F17* (família *Cyclone IV*), os circuitos de *clock*, de programação e de alimentação em uma solução pronta para uso.

Inúmeros pontos de prova e jumpers foram empregados no protótipo de modo a garantir a maior flexibilidade possível em sua avaliação.

Figura 50 – Protótipo com 8 Canais, desenvolvido como uma prova de conceito.

[17] No projeto do protótipo foram levados em conta 16 canais mas apenas 8 foram implementados.

Por fim, com o objetivo de avaliar seu desempenho, o protótipo foi levado à bancada de teste vista na Figura 51 durante nossa visita ao Instituto Nacional de Física Subatômica da Holanda – NIKHEF em maio/2016. Após diversos testes, as revisões listadas abaixo e vistas na Figura 52 mostraram-se necessárias:

A. As tensões de operação para *offset* e fundo de escala dos moduladores de amplitude não haviam sido fixadas no projeto, com possibilidade de ajuste em potenciômetro. Decidiu-se fixa-las em $2v5$;

B. O filtro passa-baixa dos moduladores de amplitude precisou ser ajustado para adequar sua resposta à corrente média drenada pelo modulador;

C. Foram incluídos capacitores de filtragem - omitidos do projeto inicial - próximos aos pinos de alimentação de diversos componentes;

D. O fator de atenuação da saída dos moduladores de pulso foi alterado para $2{,}54 \times 10^{-3}$ com uma mudança no divisor resistivo. Originalmente, este fator era $1{,}21 \times 10^{-2}$ mas concluiu-se que havia sido levada em consideração uma previsão errada de amplitude máxima do pulso.

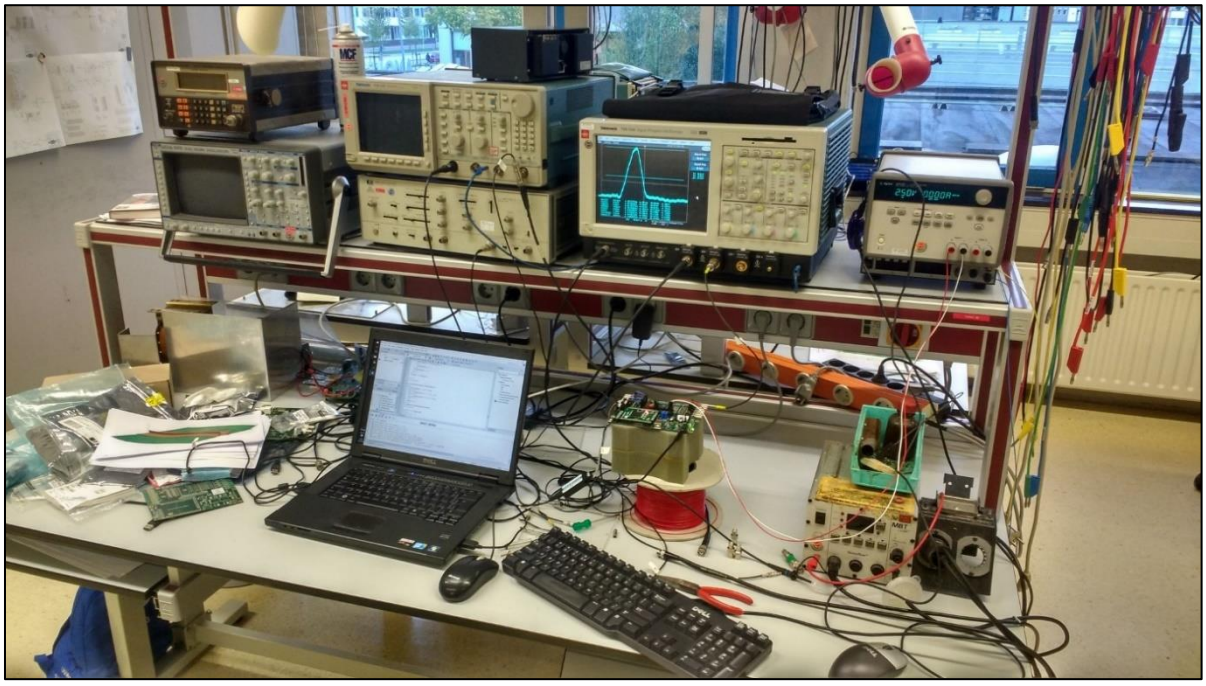


Figura 51 – Bancada de avaliação do protótipo.

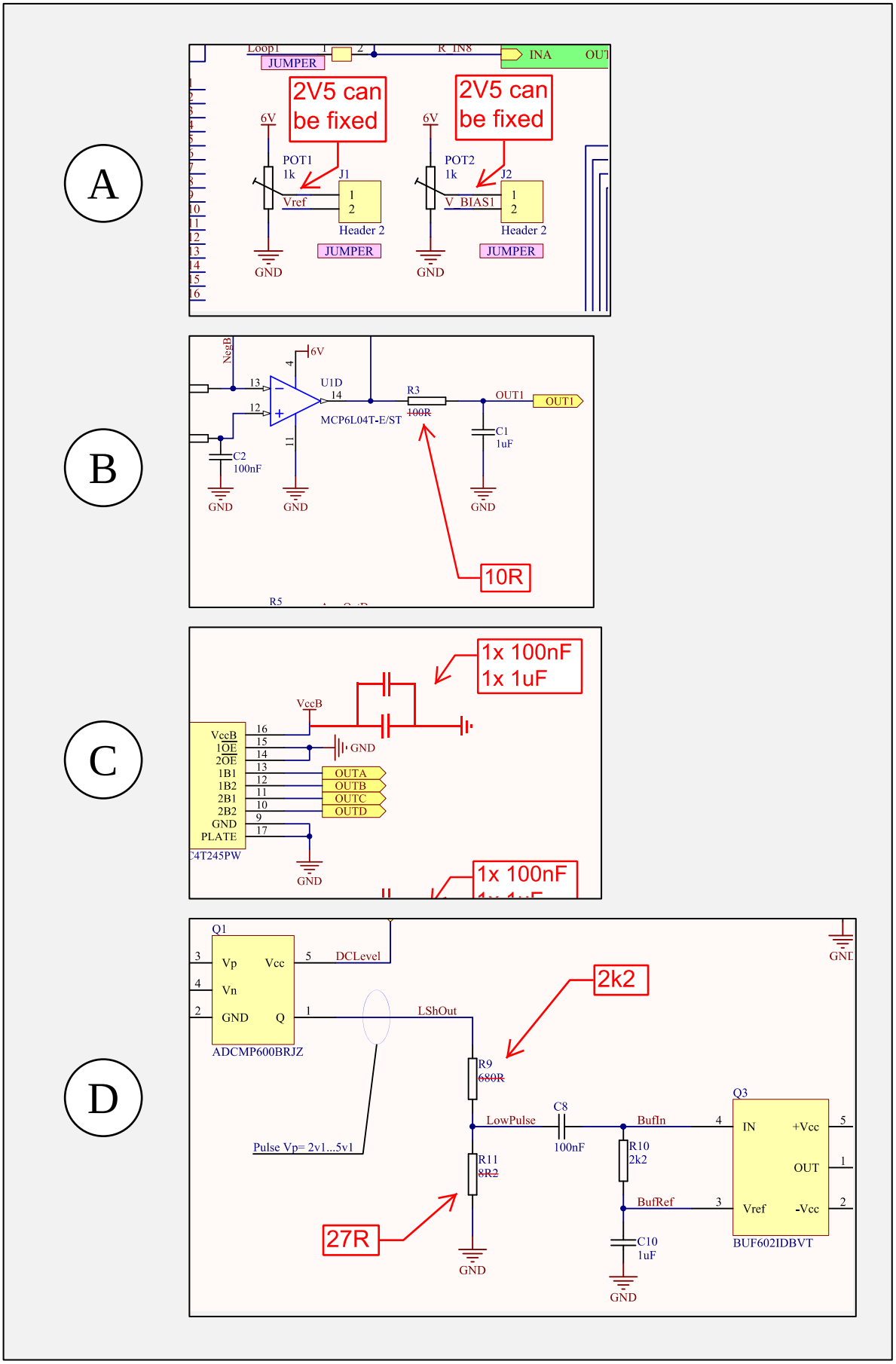


Figura 52 – Revisões após a avaliação do protótipo.

3.7. Desenvolvimento do Módulo Injetor de 256 Canais

O projeto do Módulo Injetor de 256 canais exibido na Figura 53 baseou-se no protótipo de 8 canais. Essencialmente, seu *design* baseado numa estrutura de grupos na qual os 256 canais estão divididos em 16 grupos de 16 canais cada. Na Figura 54 é possível notar a presença dos grupos, bastante evidentes no projeto da placa de circuito impresso. Cada um desses grupos teve seu *design* aproveitado a partir do protótipo revisado, como fica claro na Figura 55. Desta maneira, o projeto do Módulo Injetor foi baseado no *design* já testado e aprovado do protótipo de oito canais.

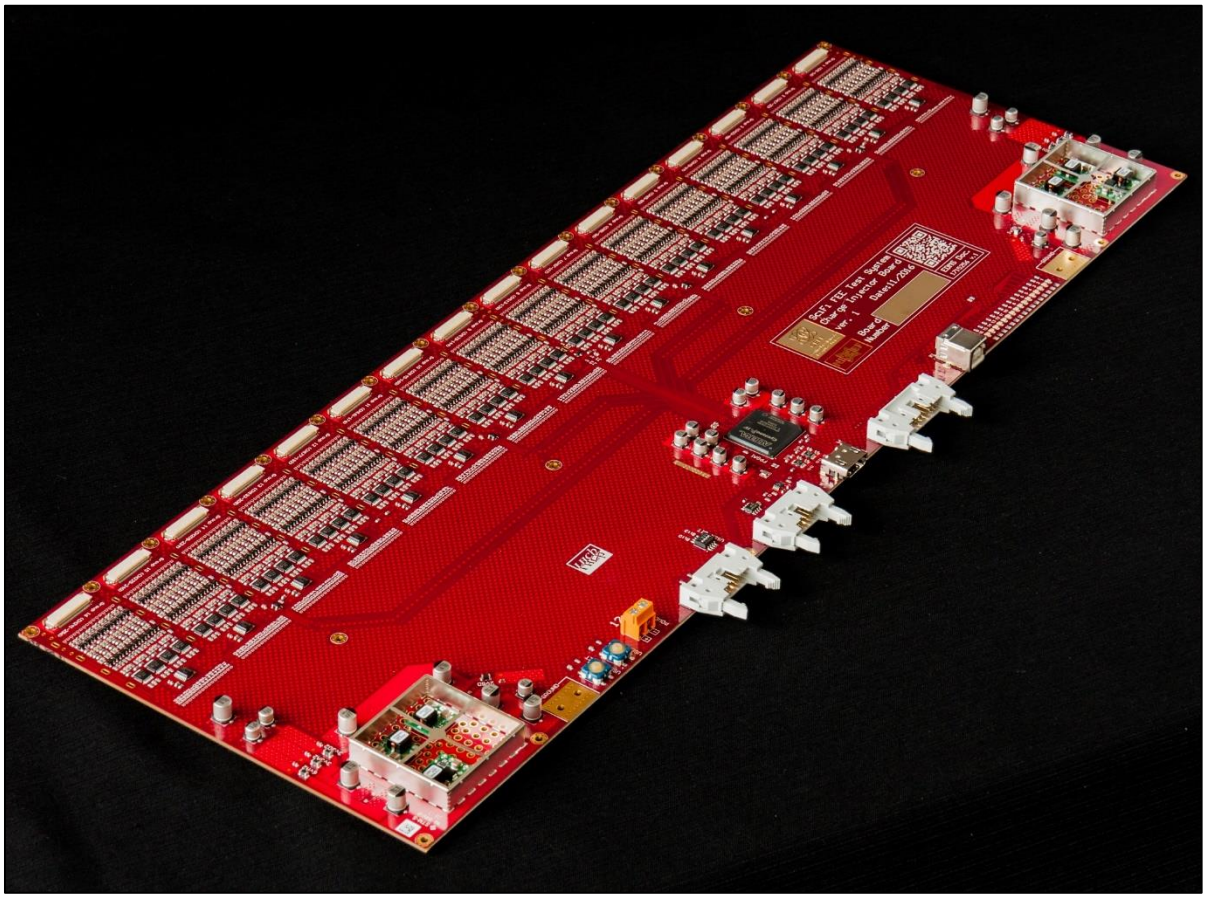


Figura 53 – Exemplar do Injetor de 256 Canais.

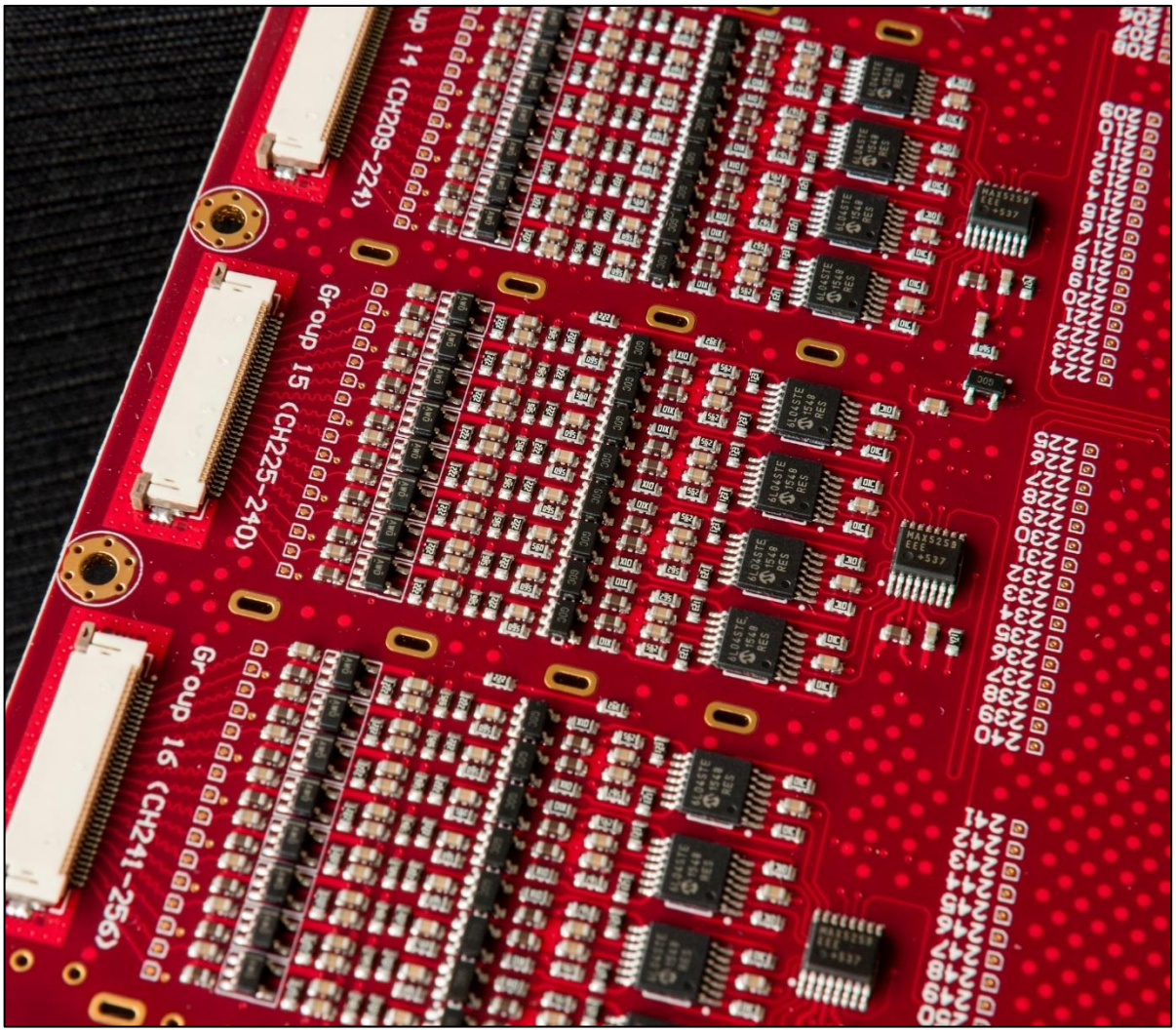


Figura 54 – Detalhe da organização em grupos do Módulo Injetor.

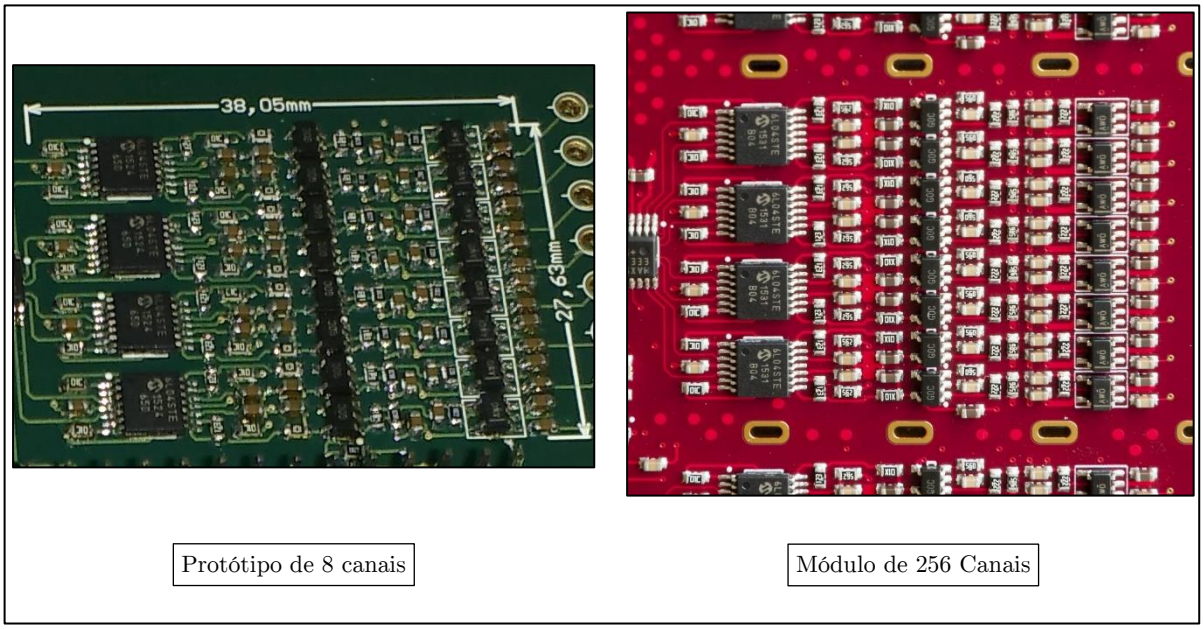


Figura 55 – Transferência de *design* a partir do protótipo.

Na Figura 56 a localização dos circuitos elétricos apresentados a seguir foi destacada sobre a placa de circuito impresso do Módulo Injetor. Parte dos componentes se encontra também na face inferior do módulo, vista na Figura 57.

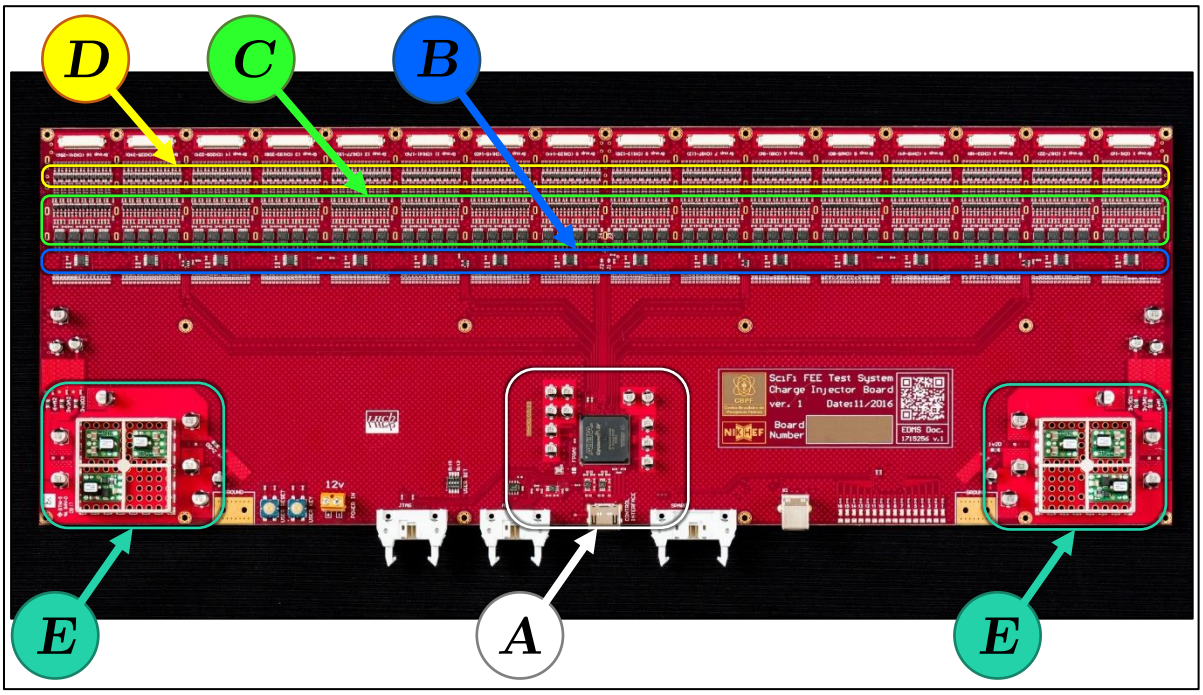


Figura 56 – Vista Superior do Injetor de 256 Canais.

A. *FPGA*, no qual estão programados os circuitos de Gerenciamento de *clock*, a Memória *LUT* e suas interfaces, os decodificadores de endereços e de controle, o gerador de pulsos e o circuito de controle dos *DAC's*. O *FPGA* e seus componentes periféricos são vistos em detalhe na Figura 58;

B. Conversores de digital para analógico;

C. Moduladores de Amplitude de Pulso;

D. Estágios de saída (*buffers*).

E. Circuito de Alimentação Elétrica, vistos em detalhe na Figura 59.

Na Figura 60 o Módulo Injetor é apresentado em um ângulo que torna evidente a magnitude desta placa de circuito impresso, que contém 5.095 componentes montados sobre uma superfície de $187{,}0\ mm \times\ 529{,}6\ mm$.

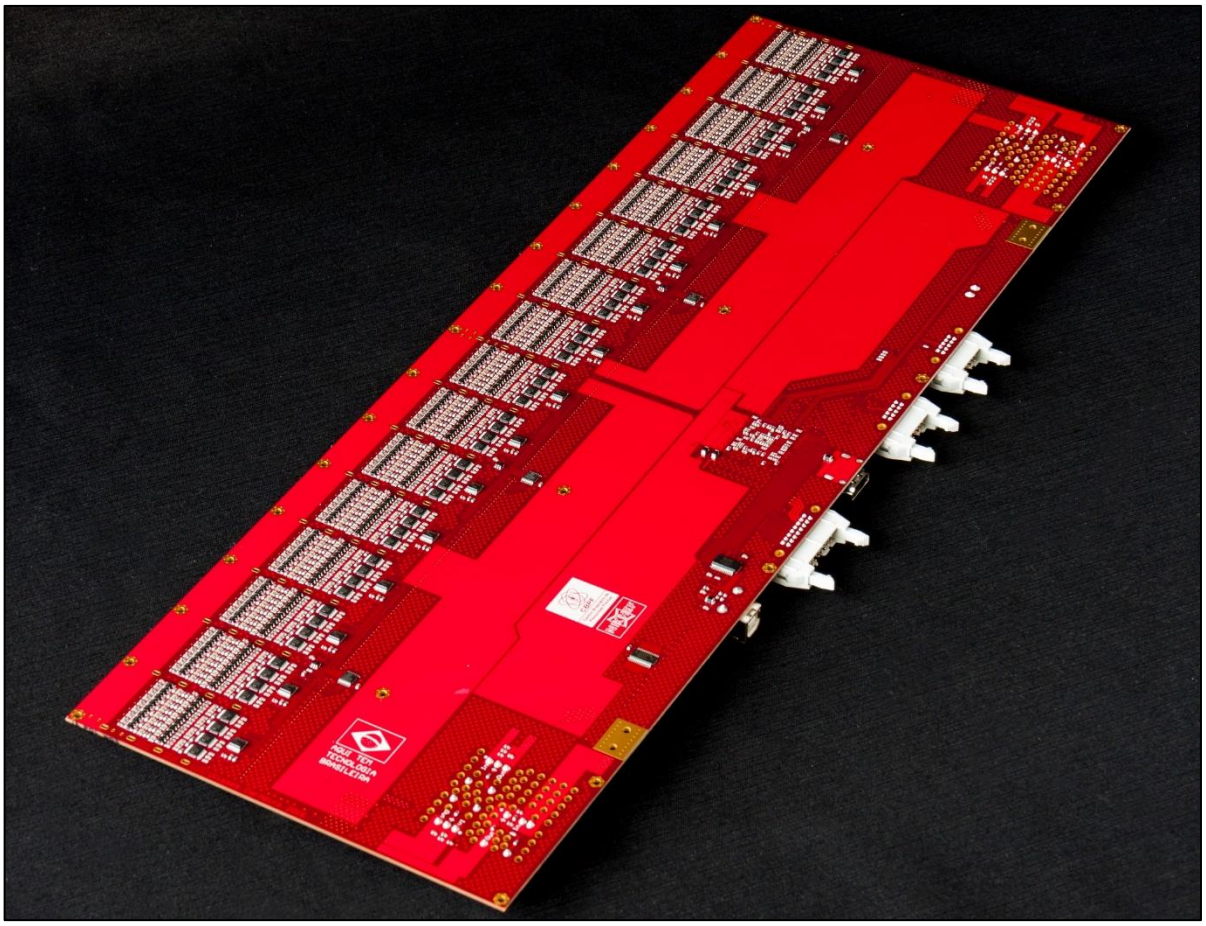


Figura 57 – Face Inferior do Módulo Injetor.

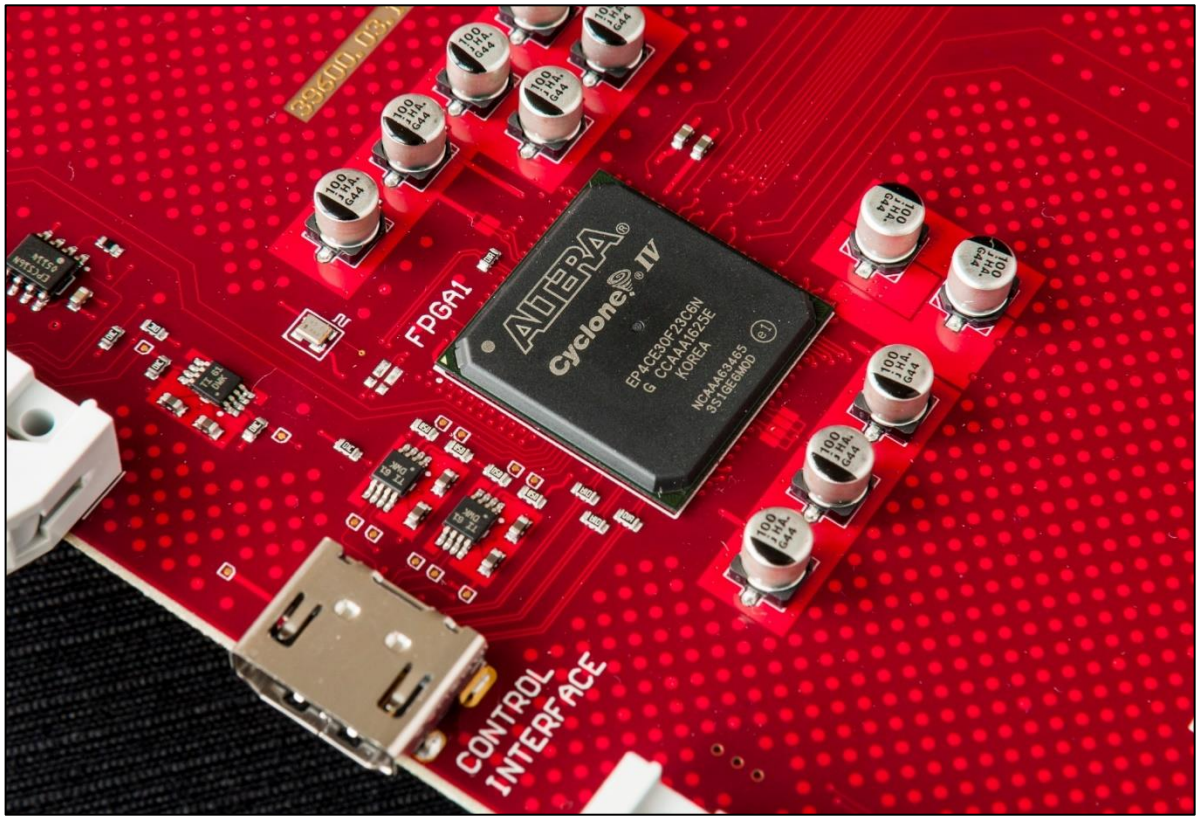


Figura 58 – Detalhe do FPGA, o concentrador do Processamento Digital.

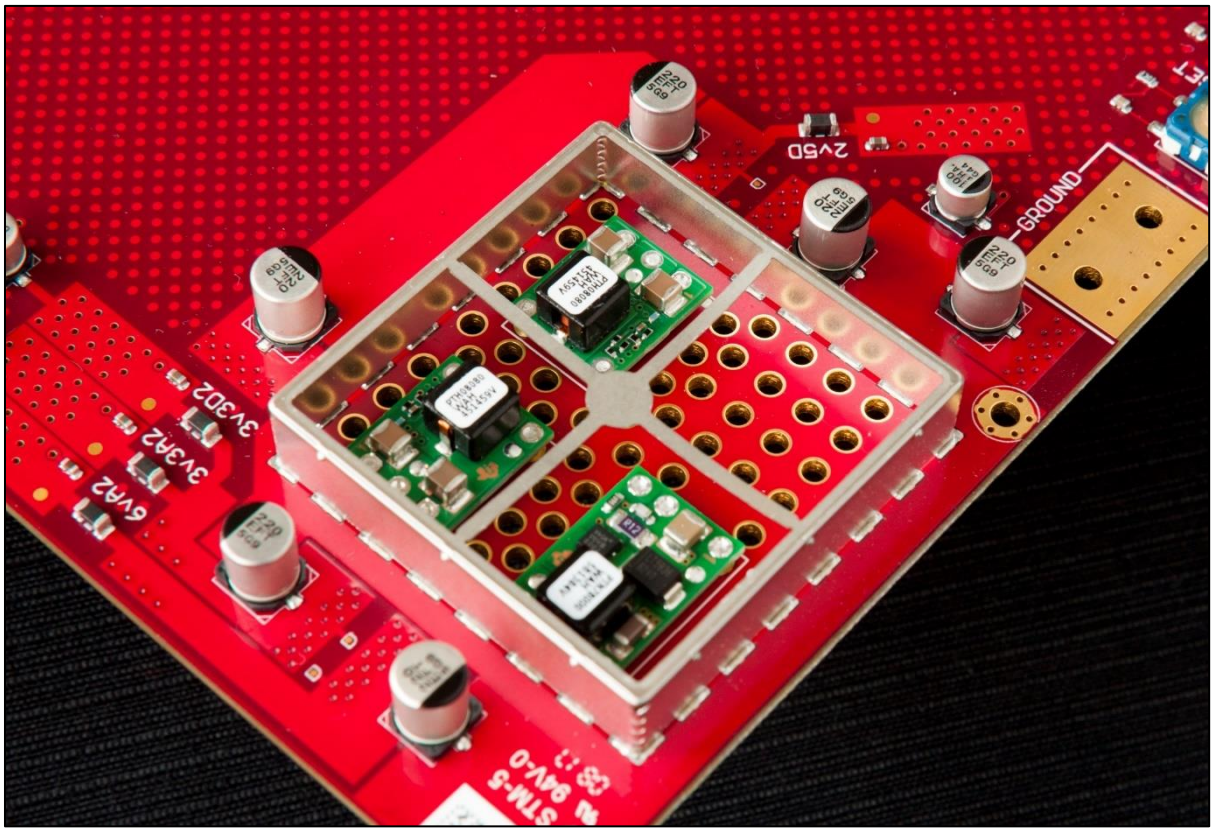


Figura 59 – Detalhe dos Circuitos Reguladores de Tensão.

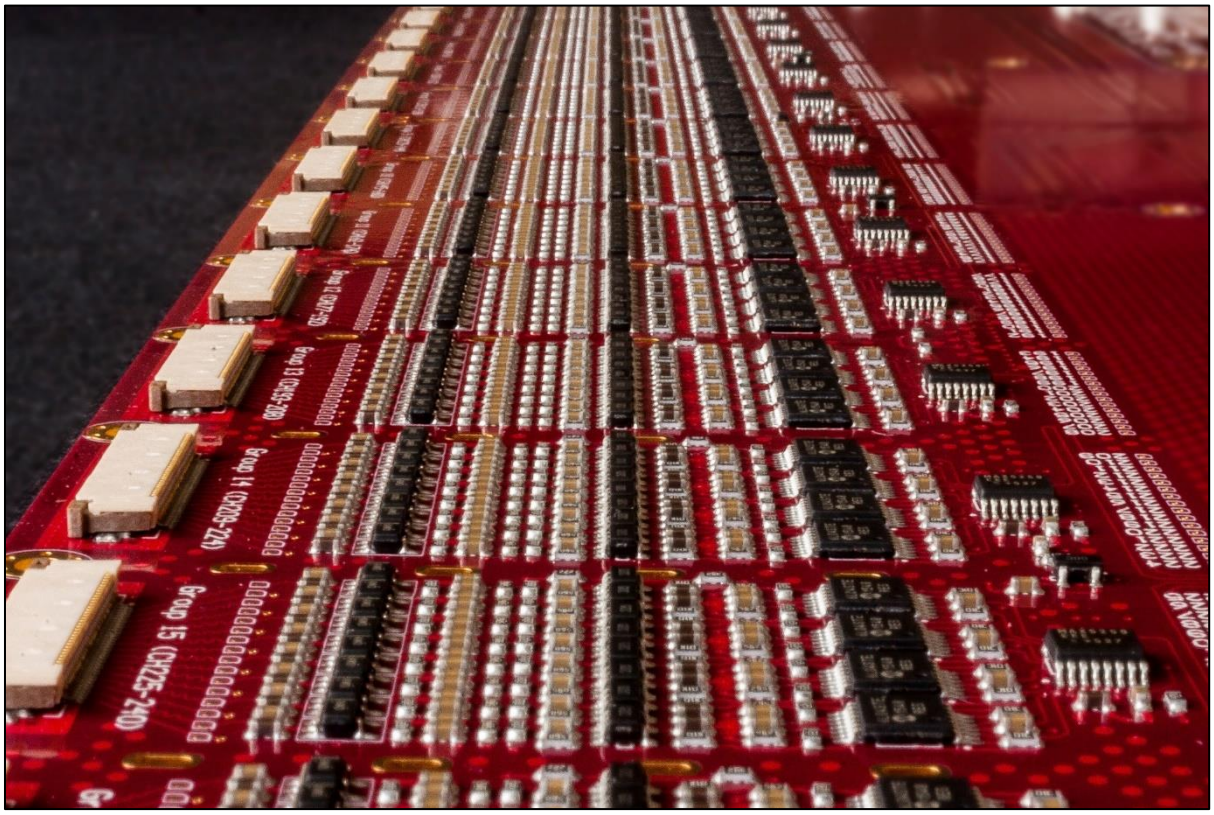


Figura 60 – Montagem dos grupos analógicos.

3.7.1. Dimensionamento das Trilhas

Um dos requisitos fundamentais para o projeto é o alinhamento temporal preciso entre os 256 canais de saída. Além disso, o Módulo Injetor possui 32 conversores analógicos controlados por linhas digitais, de 8 canais cada, responsáveis por modular as amplitudes de pulso de seus 256 canais. Os sinais destas linhas de controle precisam estar em fase para que possam ser decodificados pelos conversores.

Utilizou-se o *software* livre *PCB Toolkit* [41] para calcular a velocidade de propagação do sinal elétrico nas trilhas da placa de circuito impresso do Módulo Injetor, obtendo-se como resultado $v \approx 171\, mm/ns$. Este resultado leva em conta a constante dielétrica efetiva da placa de circuito impresso, considerando a geometria das trilhas e a banda passante do sinal.

Mostrou-se, portanto, imperativo levar em consideração os efeitos de grandes comprimentos de trilha durante o roteamento dos seguintes sinais:

- P[1..256]:
  - » Sinal de disparo de pulso, uma trilha por canal.
- 16x P_OUT[1..16]:
  - » Saídas do Injetor, uma trilha por canal.
- S[1..4], SHRSHCP, SHRSTCPP, SHRRESET e CS[1..32]:
  - » Sinais de seleção de chip (*chip select*) dos *DAC's*
- DACDIN[1..4], DACSCLK[1..2], SCLK[1..4], DACLDAC[1..2] e LDAC[1..4]:
  - » Sinais das linhas seriais de controle dos *DAC's*

Para os sinais de disparo e de saída de pulso do Módulo Injetor optou-se pelo desenho de trilhas com o mesmo comprimento, observando o menor comprimento comum possível de trilha. No caso dos sinais de disparo escolheu-se $l = 350{,}5\ mm$, como demonstrado graficamente na Figura 61 e para os sinais de saída $l = 13{,}7\ mm$, como demonstra a Figura 62.

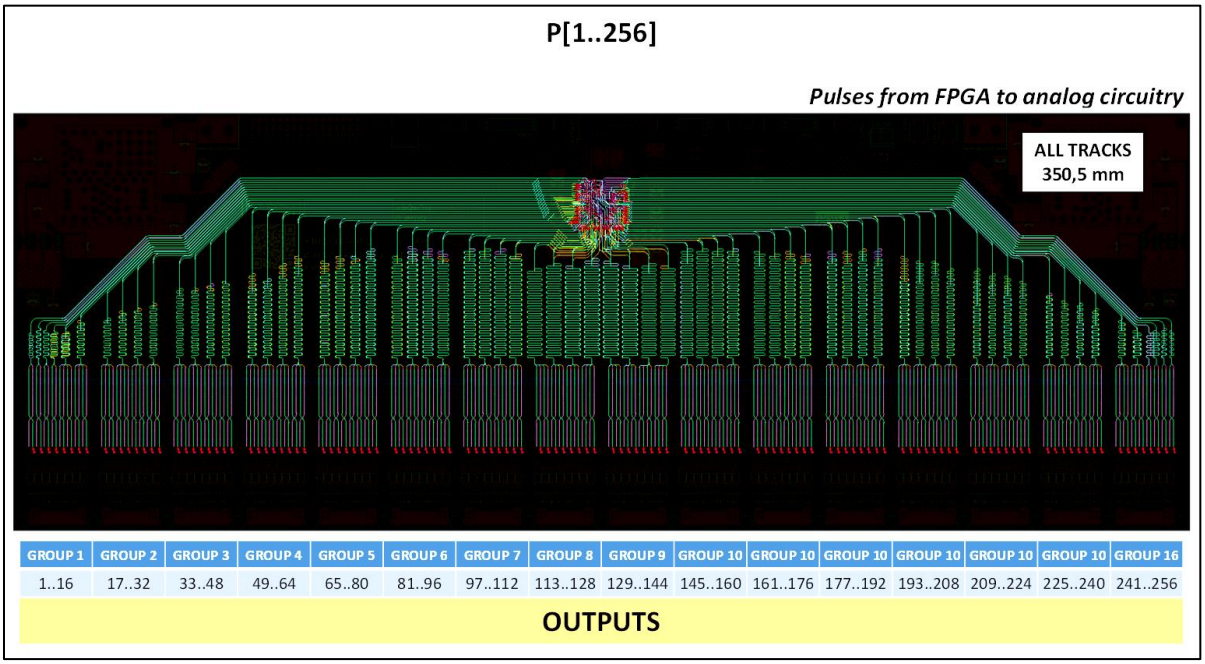


Figura 61 – Roteamento dos sinais de disparo dos canais.

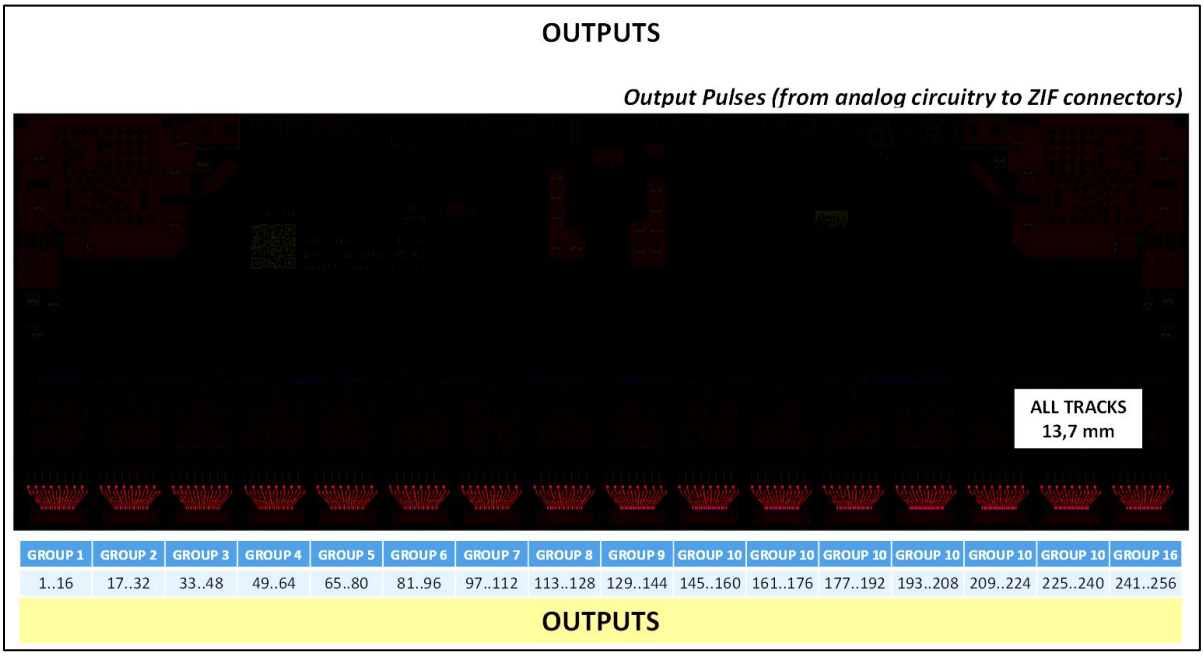


Figura 62 – Roteamento das saídas do Módulo Injetor.

Diferentemente dos sinais de disparo e de saída de pulso, distribuídos ponto a ponto, os sinais de controle para seleção dos *DAC's* são transmitidos para múltiplos receptores em ambos os lados da placa de circuito impresso. Esta topologia torna inviável a equalização absoluta obtida anteriormente. Entretanto, como estes sinais operam em frequência reduzida, é admissível certa diferença entre os comprimentos das trilhas. Desta forma, calculou-se o melhor ajuste médio de comprimento, como visto na Figura 63, na Figura 64, na Figura 65, na Figura 66 e na Figura 67 onde são demonstrados, respectivamente, os ajustes para os sinais S[1..4], SHRSHCP, SHRSTCPP, SHRRESET e CS[1..32]

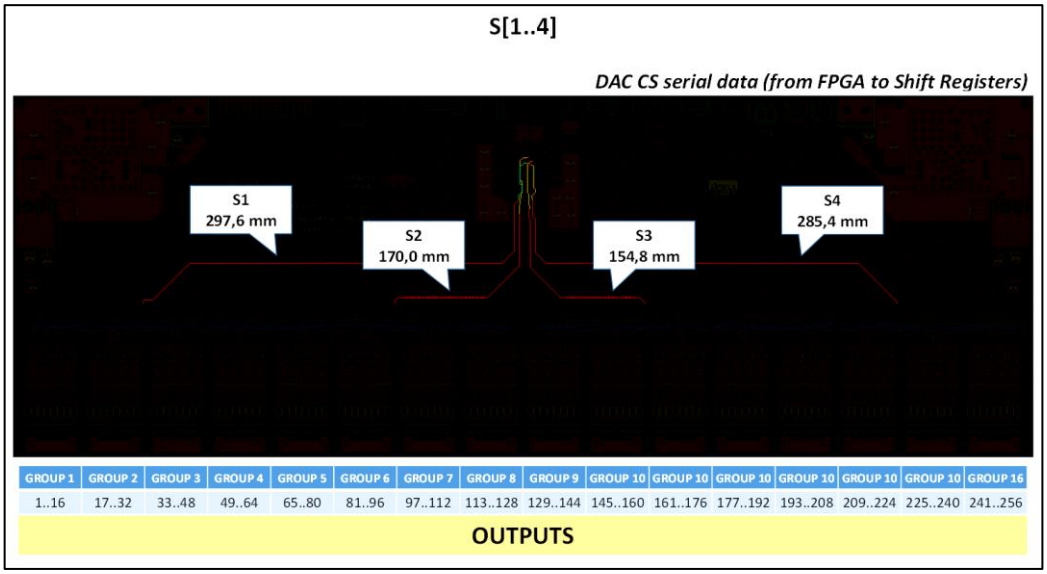


Figura 63 – Roteamento das linhas seriais de seleção dos *DAC's*.

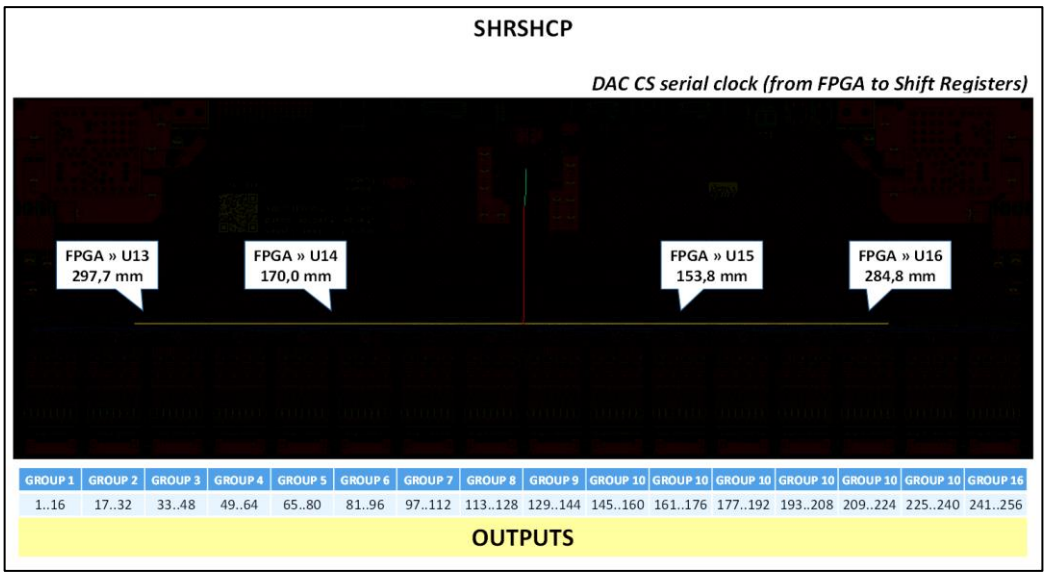


Figura 64 – Roteamento do *clock* serial para seleção dos *DAC's*.

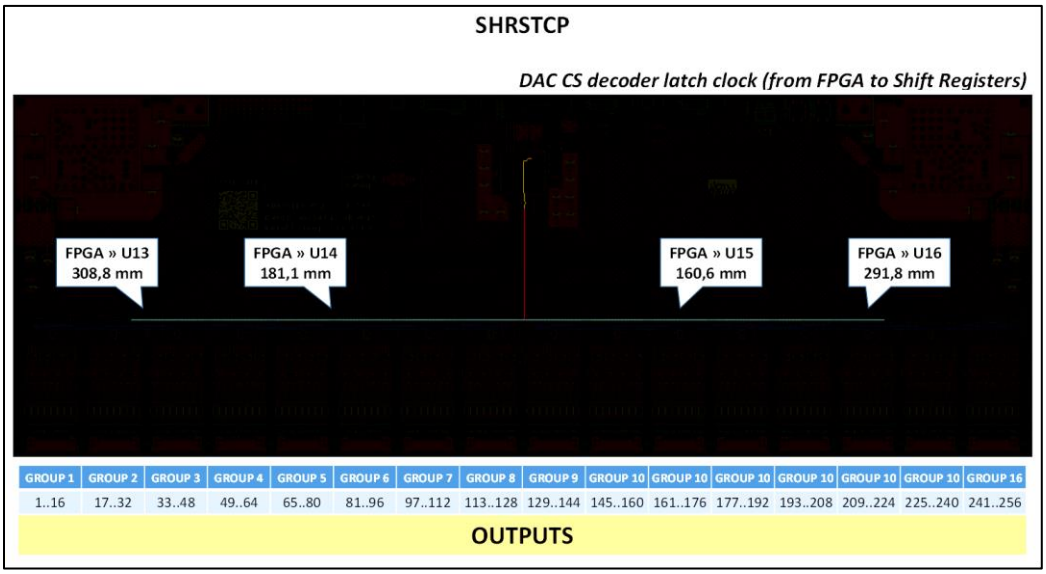


Figura 65 – Roteamento do *latch clock* para seleção dos *DAC's.*

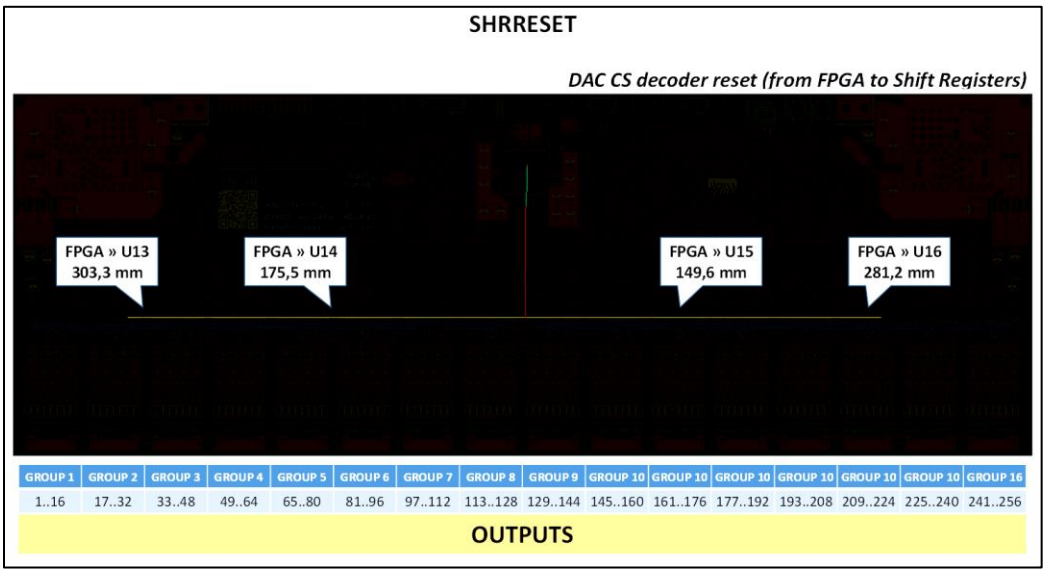


Figura 66 – Roteamento de reset dos decodificadores de seleção dos *DAC's.*

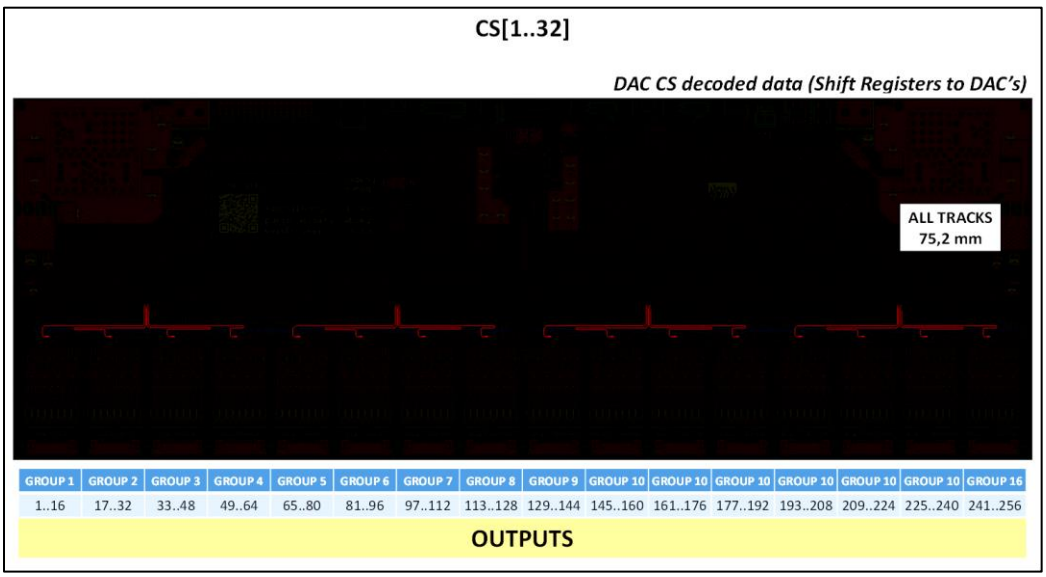


Figura 67 – Roteamento dos sinais de seleção dos *DAC's.*

Na Figura 68, a seguir, é apresentado um resumo dos comprimentos de trilha resultantes do roteamento dos sinais de seleção dos *DAC's*.

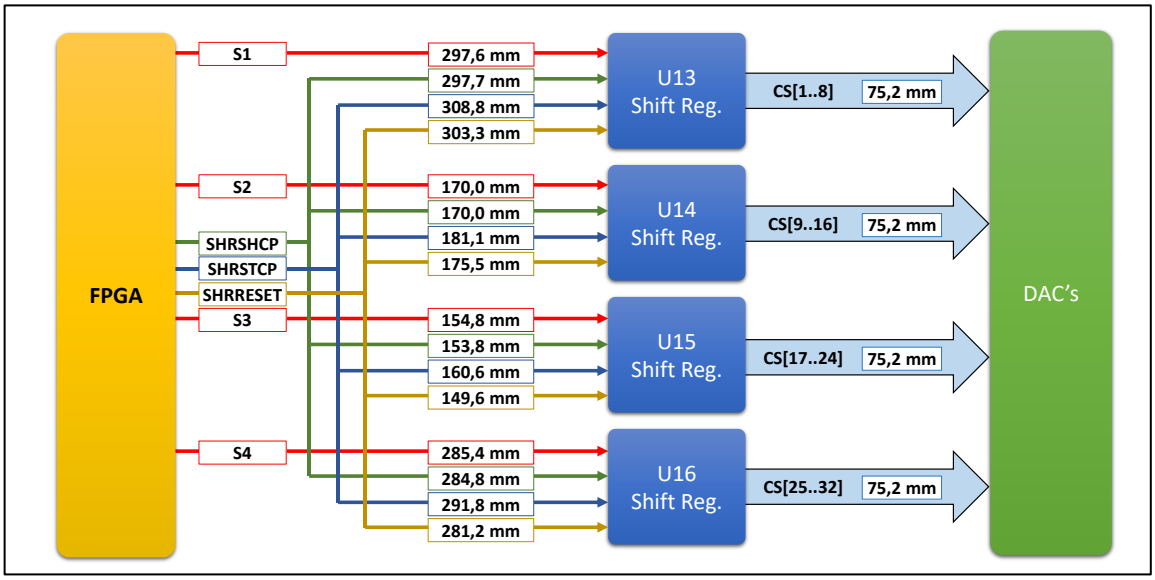


Figura 68 – Resumo do roteamento para seleção dos *DAC's*.

De maneira análoga aos sinais de seleção dos *DAC's*, os comprimentos das trilhas das linhas seriais que os controlam não podem ser equalizados absolutamente pois são distribuídos de uma fonte única para múltiplos receptores em ambos os lados do Módulo Injetor. Para estes sinais adotou-se também o melhor ajuste médio de comprimento, como visto na Figura 70, na Figura 71, na Figura 72 e na Figura 73. Na Figura 74 é apresentado um resumo dos comprimentos das trilhas de controle dos *DAC's*.

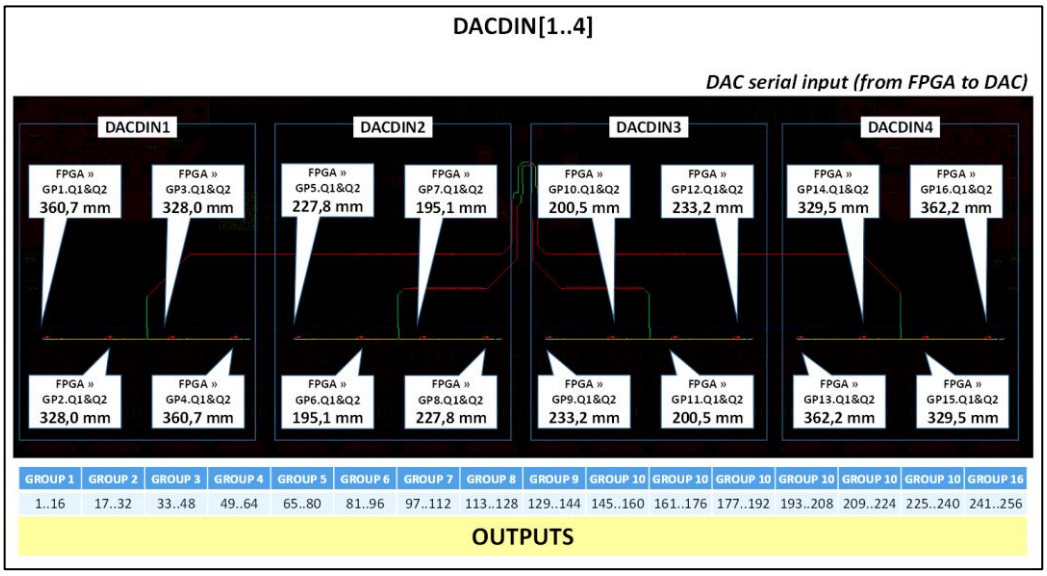


Figura 69 – Roteamento das linhas seriais de controle dos *DAC's*.

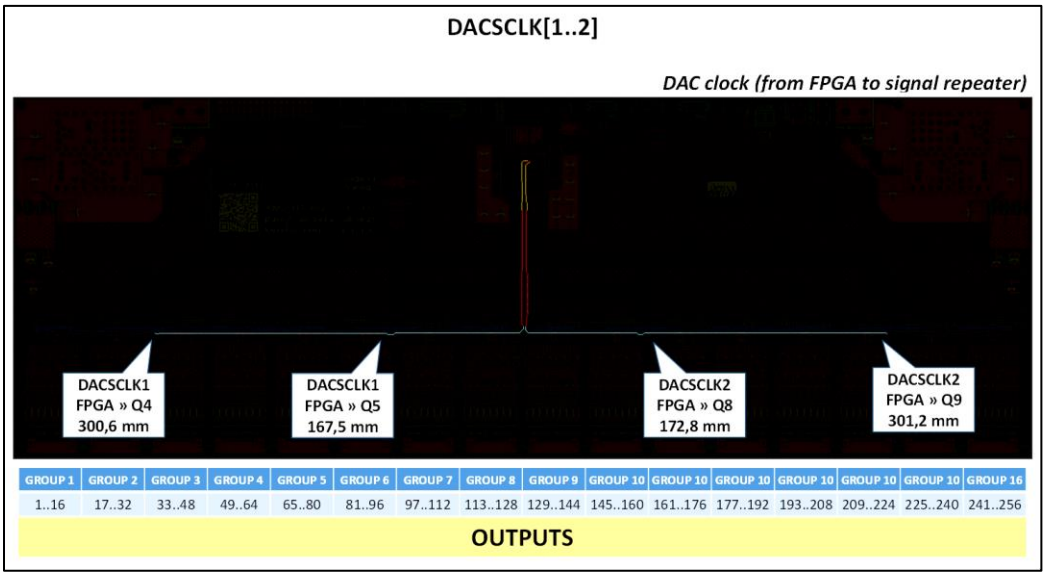


Figura 70 – Roteamento das linhas de *clock* dos *DAC's.*

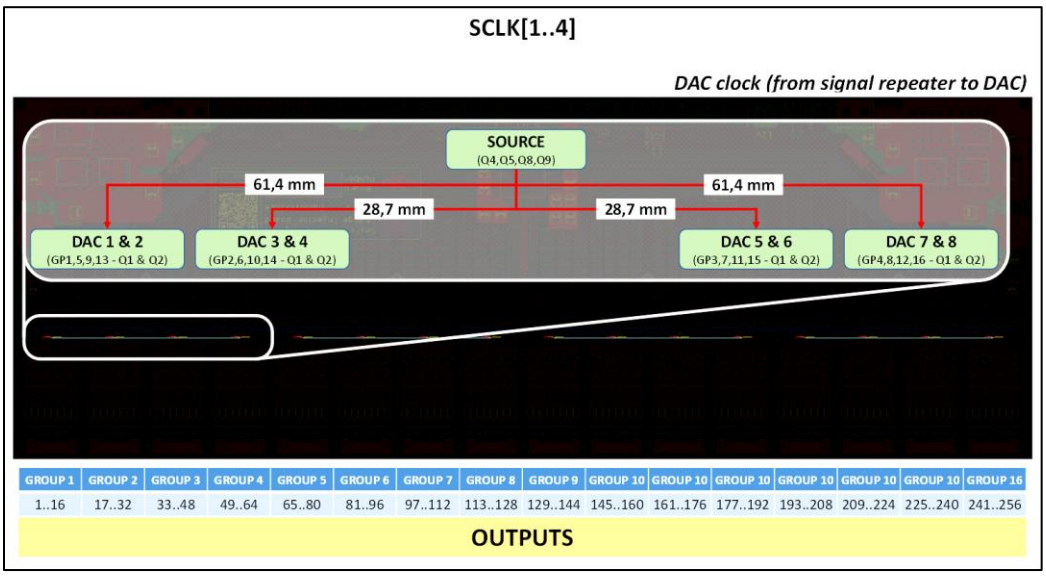


Figura 71 – Roteamento de *clock* distribuído para os *DAC's.*

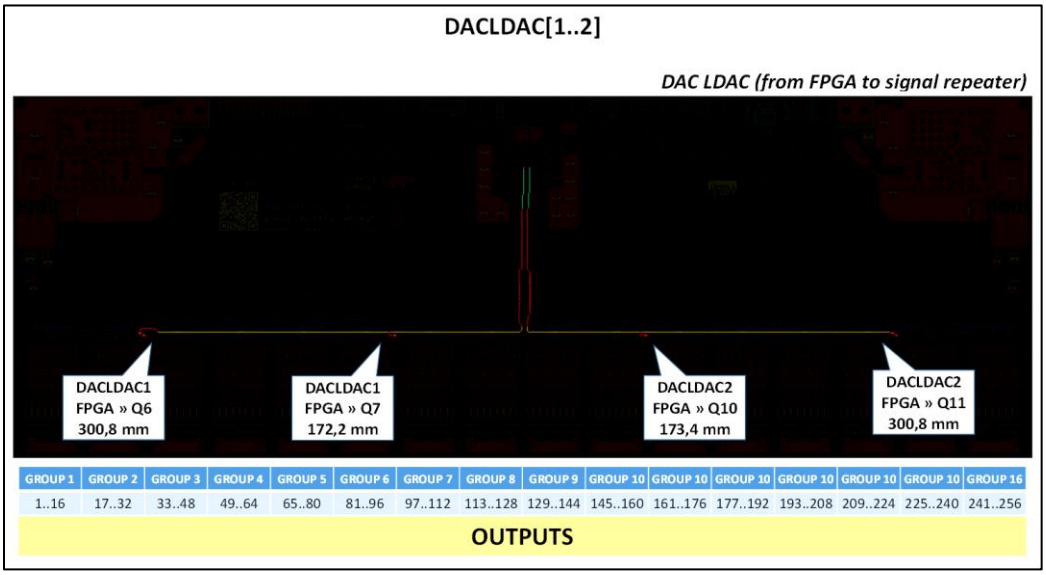


Figura 72 – Roteamento das linhas de *latch* dos *DAC's.*

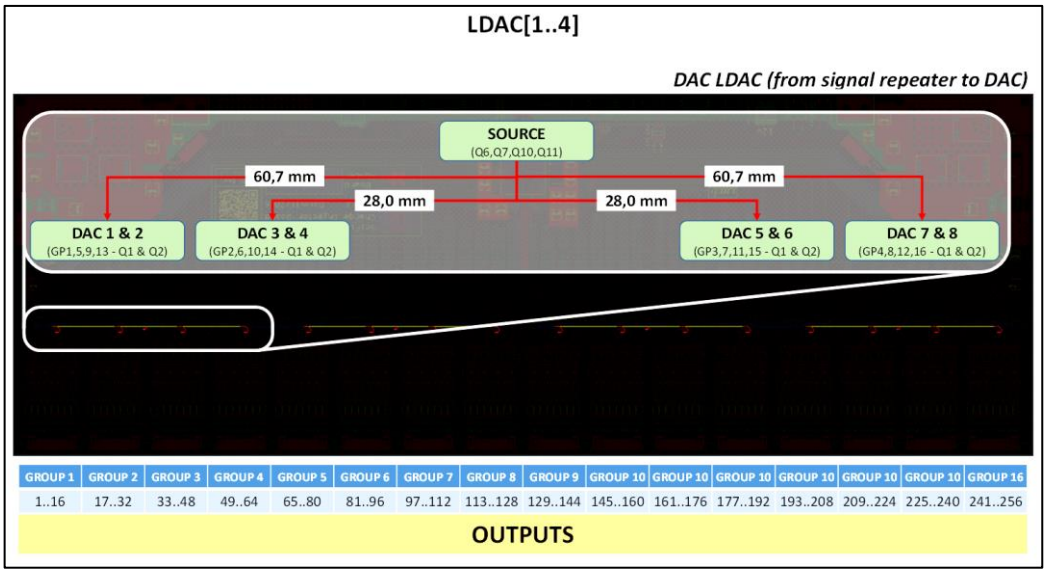


Figura 73 – Roteamento de *latch* distribuído para os *DAC's*.

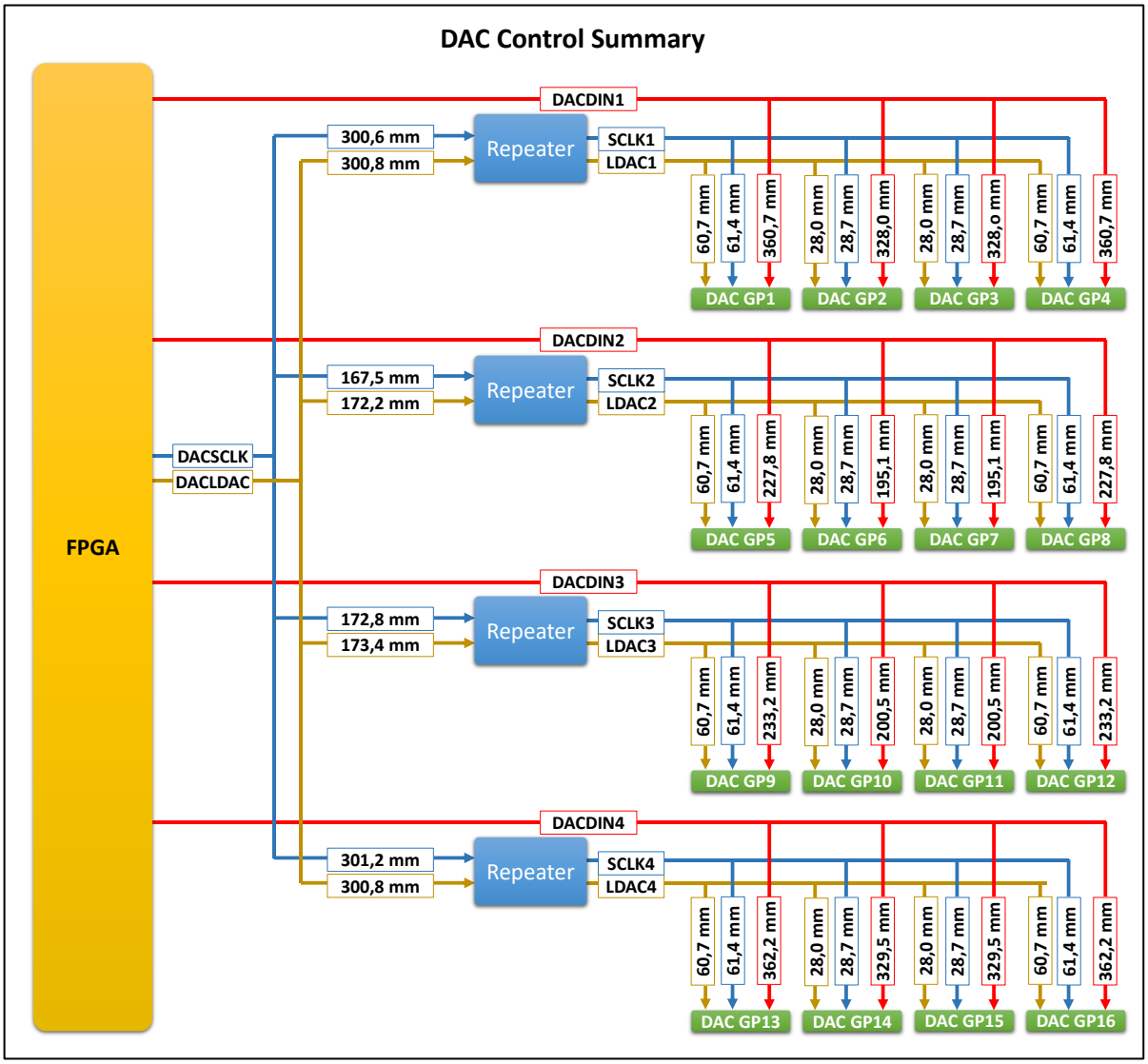


Figura 74 – Resumo do roteamento de controle dos *DAC's*.

3.8. Avaliação do Módulo Injetor de 256 Canais

Após avaliação das duas primeiras unidades produzidas foram estabelecidos os seguintes pontos de revisão do projeto para a produção das demais unidades do Módulo Injetor. Estas revisões já foram implementadas em toda a documentação que é exibida nesta dissertação.

- Correções:

  » Aumentado o isolamento do cobre próximo aos orifícios de montagem:

  Observou-se que o isolamento em torno dos orifícios de montagem estava demasiadamente pequeno. Por esta razão, tensões mecânicas causadas pelos elementos de fixação (parafusos) podem causar curto-circuito entre *ground* e planos de alimentação de camadas internas;

  » Correção do lado de baixa tensão dos conversores de nível I^2C:

  Foi constatado um erro no posicionamento de pinos dos circuitos integrados conversores de nível dos barramentos I^2C. Esta falha pôde ser corrigida manualmente nas primeiras unidades produzidas e já estará corrigida nas próximas unidades;

- Melhorias:

  » Adicionados pontos de *GND* próximo aos pontos de teste de pulso;

  » Incluído um conector *LEMO* como alternativa para o sinal *SP2v5-1.*

  Estas melhorias foram inspiradas nas dificuldades encontradas durante o procedimento de teste dos Módulos Injetores e visam facilitar a execução deste procedimento nas próximas unidades produzidas.

3.9. Controle Lento do Módulo Injetor

Como introduzido na Seção 3.4, o controle dos Módulos Injetores objetiva controlar a quantidade de carga por pulso injetado através da manipulação da largura e da amplitude destes pulsos. Estes parâmetros, relacionados à quantidade de carga por pulso, foram considerados de baixa velocidade e compõem o chamado controle lento (ou "*slow-control*") do Módulo Injetor.

O controle lento se dá pela manipulação de registradores armazenados no banco de registradores do *FPGA*, que pode ser acessado por diferentes meios:

- Interface *USB*: Incluída no projeto do *hardware* para controle do Módulo Injetor através da emulação de uma porta serial em um PC. Esta interface encontra-se totalmente implementada em *hardware*, mas sua implementação de *software* não foi concluída e é sugerida como continuidade do projeto;

- Interface I^2C de *debug*: Esta interface é acessível através dos pinos 9 e 10 do conector *SPARE* do Módulo Injetor. Destina-se ao controle avulso – em modo *standalone* – do Módulo (com leitura e escrita) ou à monitoração dos registradores (apenas leitura) quando há um Módulo de Controle conectado;

- Interface I^2C para o Módulo de Controle: Esta interface é acessível através dos pinos 15 e 16 no conector *HDMI* do Módulo Injetor e destina-se ao controle integrado do módulo, isto é, quando um Módulo de Controle está conectado e controla vários Módulos Injetores.

Além de permitir o controle lento do Módulo Injetor o banco de registradores armazena os cenários para controle rápido, que será melhor descrito a seguir. É importante notar que mais de um meio pode ser simultaneamente utilizado para leitura dos registradores, mas que apenas um deve ser escolhido para escrita. A Switch S3 (USER BIT) do Módulo Injetor permite a seleção da interface habilitada para escrita.

3.9.1. Adaptador *I2C - USB*

Um adaptador *I²C-USB* foi desenvolvido para possibilitar o controle de um Módulo Injetor por um microcomputador através da interface *I²C* de *debug*. O adaptador, exibido na Figura 75 abaixo, consiste em um *Arduino Mini Pro* conectado a um conversor *USB/UART TTL* baseado no circuito integrado *FTDI*. No microcontrolador *ATmega* do *Arduino* foi gravado o código reproduzido no Apêndice E que permite ao usuário ler e escrever valores no banco de registradores do Módulo Injetor através da interface de usuário em terminal serial vista na Figura 76.

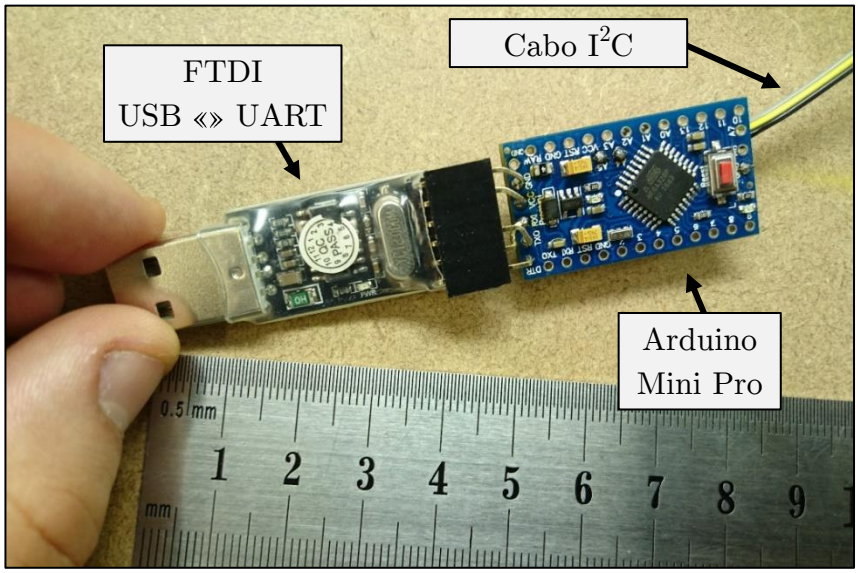


Figura 75 – Adaptador *I²C/USB* desenvolvido para controle do Módulo Injetor.

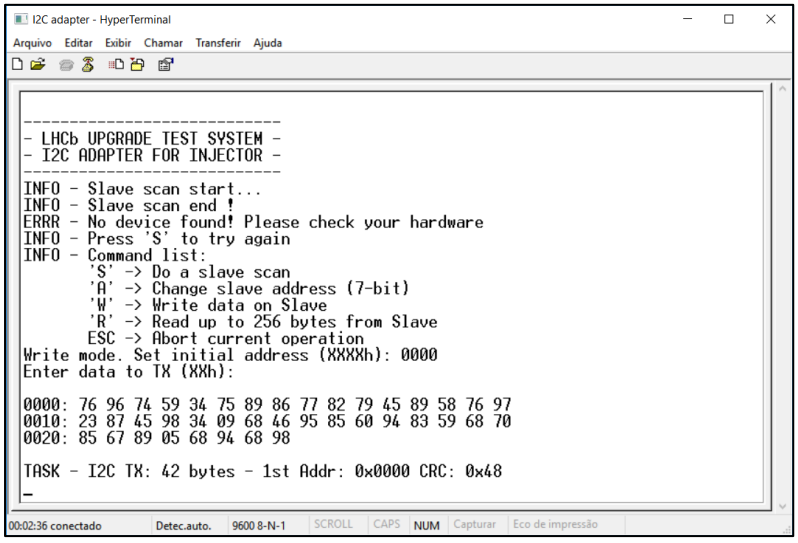


Figura 76 – Interface de usuário do adaptador *I²C/USB* no terminal serial.

3.9.2. Controle pelo Sistema SCADA[18]

A configuração proposta para o Dispositivo Injetor requer controle simultâneo de múltiplos Módulos Injetores. A ferramenta apresentada na Seção 3.9.1 executa controle manual de apenas um Módulo Injetor com propósito de teste, não sendo adequada para o Dispositivo Injetor de Carga. Para esta finalidade aproveitou-se da compatibilidade do sistema *GBTx* com o barramento I^2C e foi previsto um caminho de controle lento através de um chip *GBT-SCA* no Módulo de Controle. Este caminho está demonstrado no diagrama em blocos da Figura 77. Desta forma, cada Módulo do Dispositivo Injetor do sistema de teste pode ser controlado individualmente através de uma linha I^2C exclusiva, cujo Master é o chip *GBT-SCA* no Módulo de Controle.

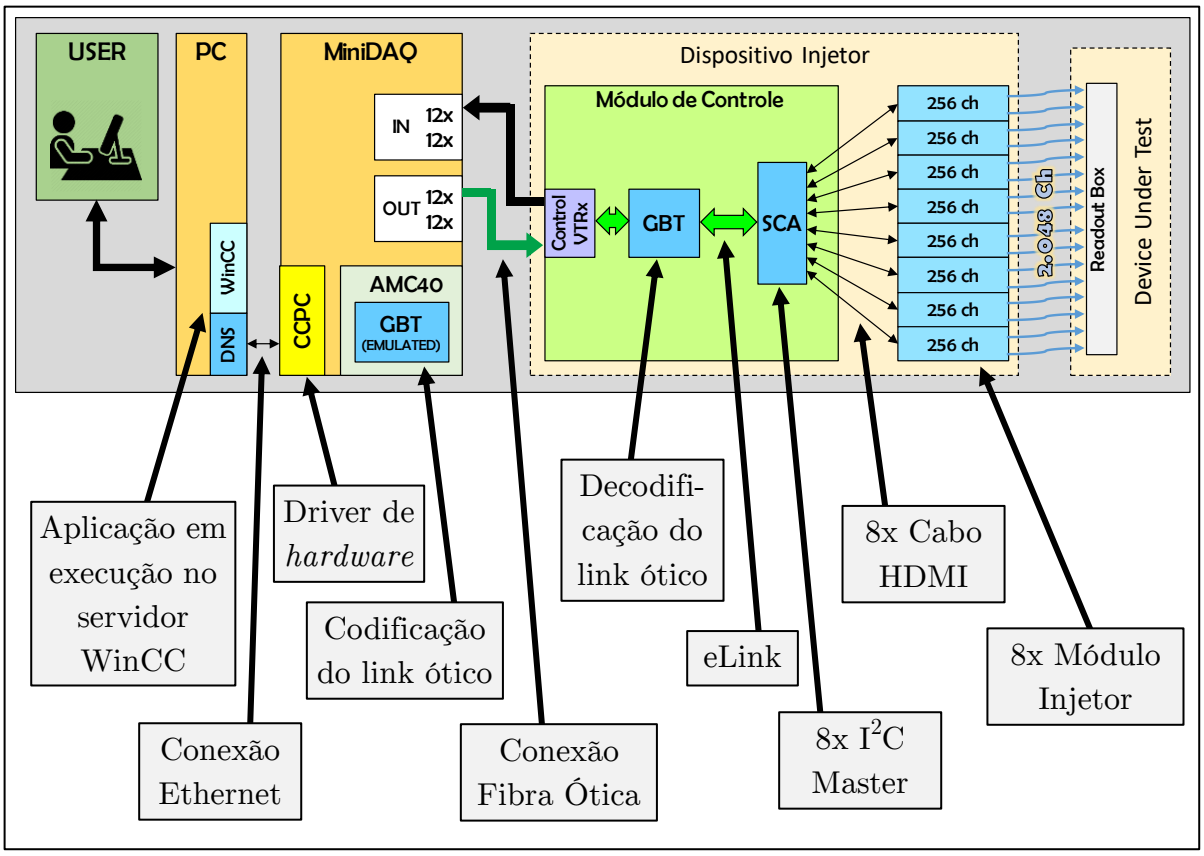


Figura 77 – Diagrama em blocos do caminho de controle pelo sistema SCADA.

[18]Sigla em inglês para *Supervisory Control and Data Acquisition System*. Sistemas *SCADA* baseados no *software WinCC-OA* são amplamente utilizados para o controle dos experimentos no CERN.

O desenvolvimento de uma ferramenta *SCADA* completa para controle do sistema de teste não faz parte do escopo deste trabalho. Entretanto, foi desenvolvida uma aplicação simples em ambiente *WinCC-OA* apenas para validar o funcionamento deste caminho de controle. Através do painel visto na Figura 78 foi possível configurar manualmente os parâmetros de pulso do par Canal / Módulo Injetor selecionado.

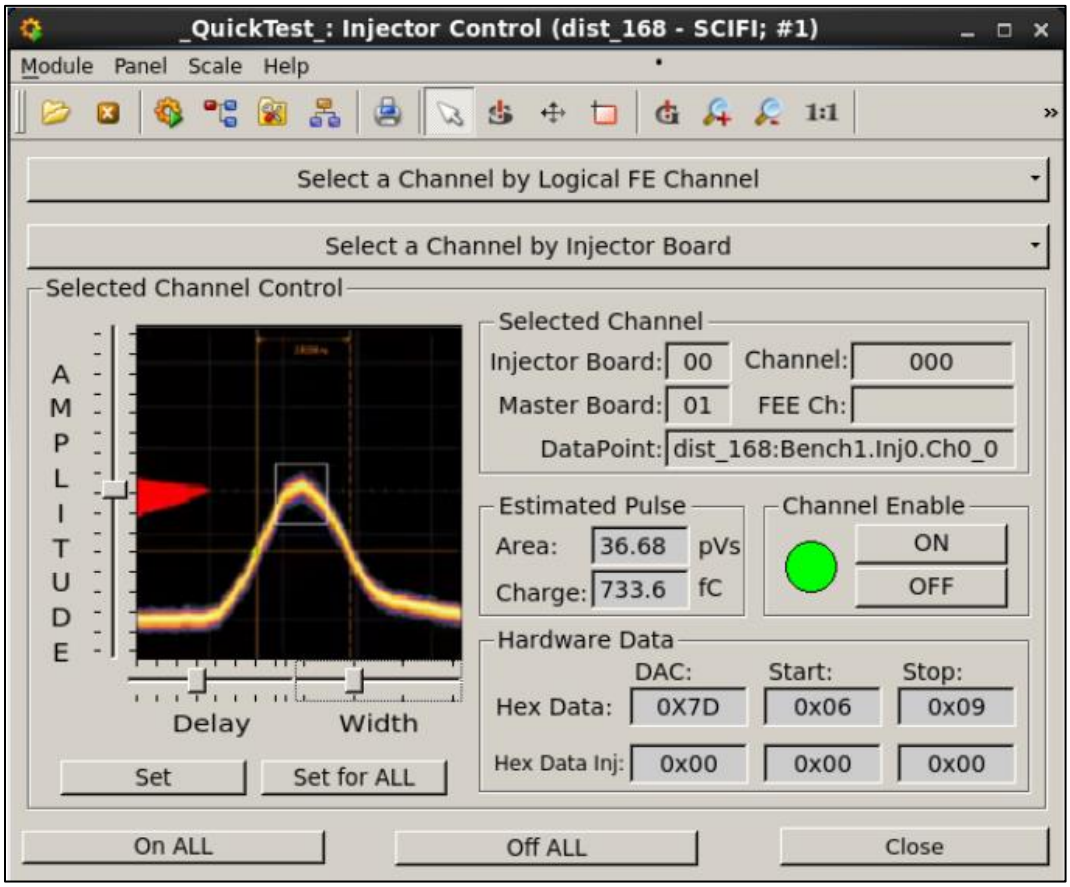


Figura 78 – Interface de usuário da aplicação SCADA programada no WinCC.

3.10. Controle Rápido do Módulo Injetor

A formação dos clusters descritos em 2.8.3 no *readout* das estações *SciFi* requer um controle preciso do *timing* de injeção e a possibilidade de ligar e de desligar à taxa plena de $40MHz$ individualmente os canais do Dispositivo Injetor. Parâmetros relacionados a estes ajustes foram considerados de alta velocidade e seu controle compõe o então chamado controle rápido (ou "*fast-control*") do Módulo Injetor.

O caminho de controle rápido consiste em duas linhas diferenciais[19] entre os Módulos de Controle e Injetor. Estas linhas transmitem a $160\,Mbps$, ou seja, disponibilizam quatro bits para cada janela de disparo de $25\,ns$. Decodificado, este fluxo permite selecionar à taxa plena de $40MHz$ um dentre os $2^4 = 16$ cenários de injeção.

O cenário de injeção é uma região previamente carregada através de uma das interfaces de controle lento no banco de registradores. Cada cenário contém configurações de *timing* e de estado (ligado/desligado) para todos os canais do Módulo Injetor. Um cenário diferente pode ser selecionado a cada janela de disparo para que se obtenha o controle rápido exigido. O mecanismo é ilustrado no diagrama da Figura 79.

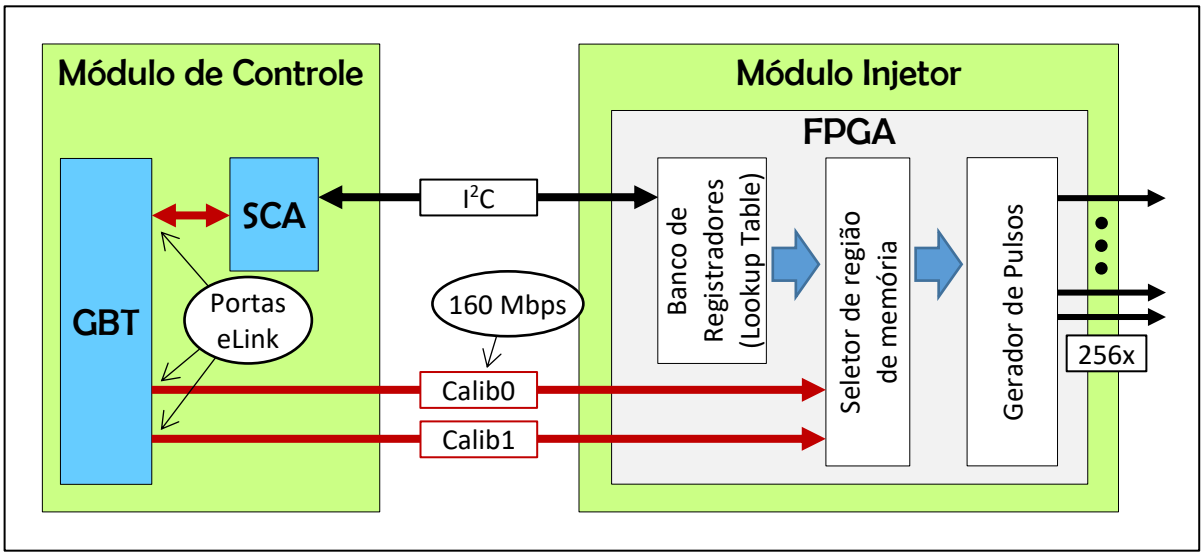


Figura 79 – Diagrama em blocos do mecanismo de controle rápido.

[19]Pares diferenciais *Calib1* e *Calib2*, acessíveis através dos pinos 1-3 e 4-6 do conector *HDMI*

3.11. Módulo de Controle

O Módulo de Controle, visto na Figura 80, tem a finalidade de distribuir para até oito Módulos Injetores os sinais de controle lento, rápido e de temporização oriundos de uma das interfaces ópticas do MiniDAQ. Foi projetado e construído em colaboração com o Instituto Nacional de Física Subatômica da Holanda – NIKHEF, parceiro do CBPF neste projeto. Este módulo é essencialmente uma versão reduzida da *Master Board* (descrita em 2.8.4) na qual foram colocados conectores HDMI no lugar de oito chips GBT serializadores de dados. Desta maneira, os Módulos Injetores se integram à arquitetura de controle já desenvolvida para o experimento de maneira quase transparente e, do ponto de vista do *hardware*, são integrados como se fossem os GBT's serializadores da *Master Board*.

Figura 80 – Módulo de Controle do Dispositivo Injetor de Carga.

3.12. Montagem Mecânica

Para finalizar o Dispositivo Injetor apresentado na Figura 40 será necessário integrar os oito Módulos Injetores ao Módulo de Controle e construir uma estrutura que permita o fácil acoplamento das Caixas de Leitura a serem testadas.

Esta estrutura já se encontra projetada, mas ainda não foi fabricada. Na Figura 81 apresenta-se uma vista do projeto com a Caixa de Leitura acoplada e na Figura 82 os componentes da montagem são destacados.

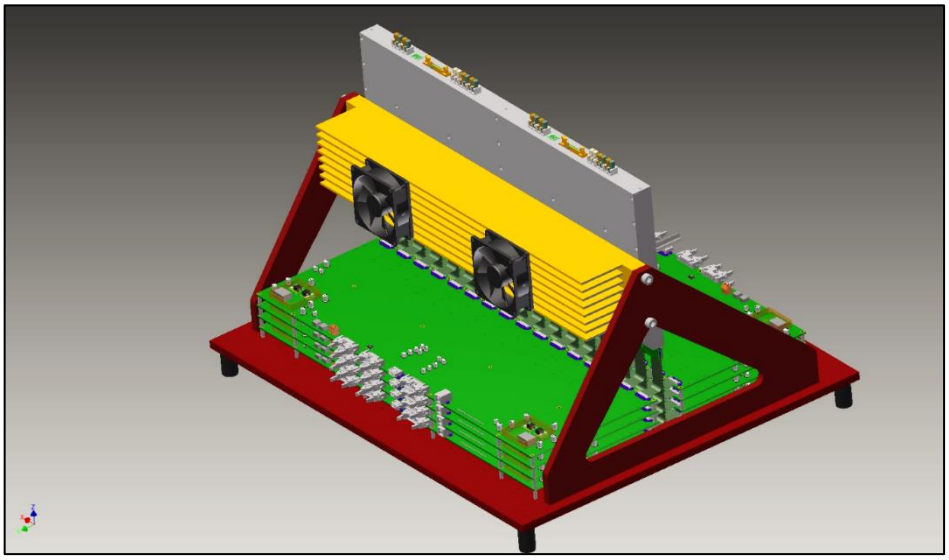


Figura 81 – Vista tridimensional do projeto da montagem mecânica.

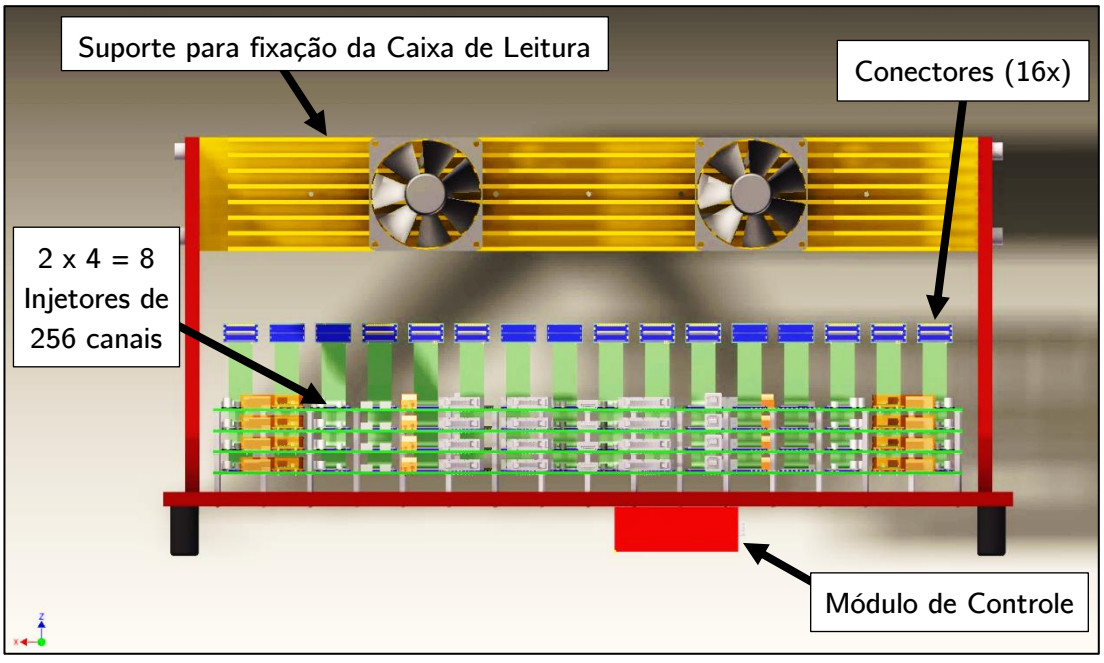


Figura 82 – Vista frontal da montagem mecânica.

Capítulo 4
Resultados Obtidos

Este capítulo apresenta os resultados das várias medições realizadas em duas ocasiões, com exemplares distintos do Módulo Injetor de Carga.

4.1. Primeira Sequência de Testes

A primeira sequência de testes foi realizada em maio de 2017 no *Nikhef* (Amsterdam) com o exemplar Número de Série #1 do Módulo Injetor com o objetivo de avaliar preliminarmente a resposta do *ASIC* **PACIFIC** versão ***4b*** a várias configurações de pulsos injetadas. Avaliou-se, também preliminarmente, alguns parâmetros de calibração do Módulo Injetor.

4.1.1. Ferramentas e Metodologia

Todas as medidas foram processadas estatisticamente no osciloscópio *Keysight* modelo *DSO-S 254A* que operou adquirindo, em momentos distintos, o sinal de saída do Módulo Injetor através de acoplamento direto (terminação interna de $50\,\Omega$) e o sinal das portas analógicas de *DEBUG* do **PACIFIC** através de sondas ativas *Agilent N2795A* $10\,M\Omega$. A conexão destas sondas ativas aos pinos de *DEBUG* é vista na Figura 84 e na Figura 85 e mostrou-se bastante crítica. Para esta sequência de testes, o Módulo Injetor foi sincronizado através de uma saída *deskewable clock* ($40MHz$) do chip *GBTx* da *Control Board*. Uma varredura de atraso foi executada com o objetivo de ajustar a fase do pulso injetado para a maior fração de carga na janela de integração do **PACIFIC**. O FPGA do Módulo Injetor foi programado para gerar um disparo de pulso a cada 100

ciclos de *clock*, ou seja, disparando a $400kHz$. Um ponta de prova para *trigger* do osciloscópio foi conectada a outro canal do Módulo Injetor, programado para fornecer uma cópia atrasada do sinal de disparo ($400kHz$). Esta técnica de *trigger* foi adotada para eliminar as interferências provocadas pela ponta de prova. Os registros estatísticos baseiam-se em pelo menos 10 mil amostras na avaliação da saída do injetor e de mil amostras na avaliação das portas analógicas de *DEBUG* do **PACIFIC**.

A bancada utilizada nesta sequência de testes é vista na Figura 83.

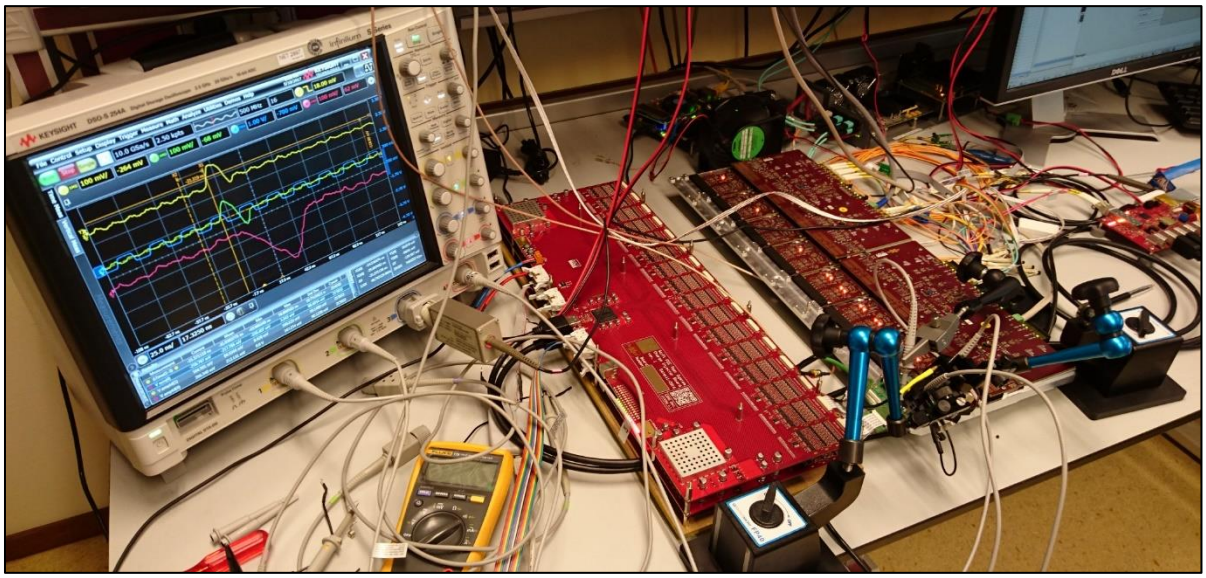


Figura 83 – Bancada de testes.

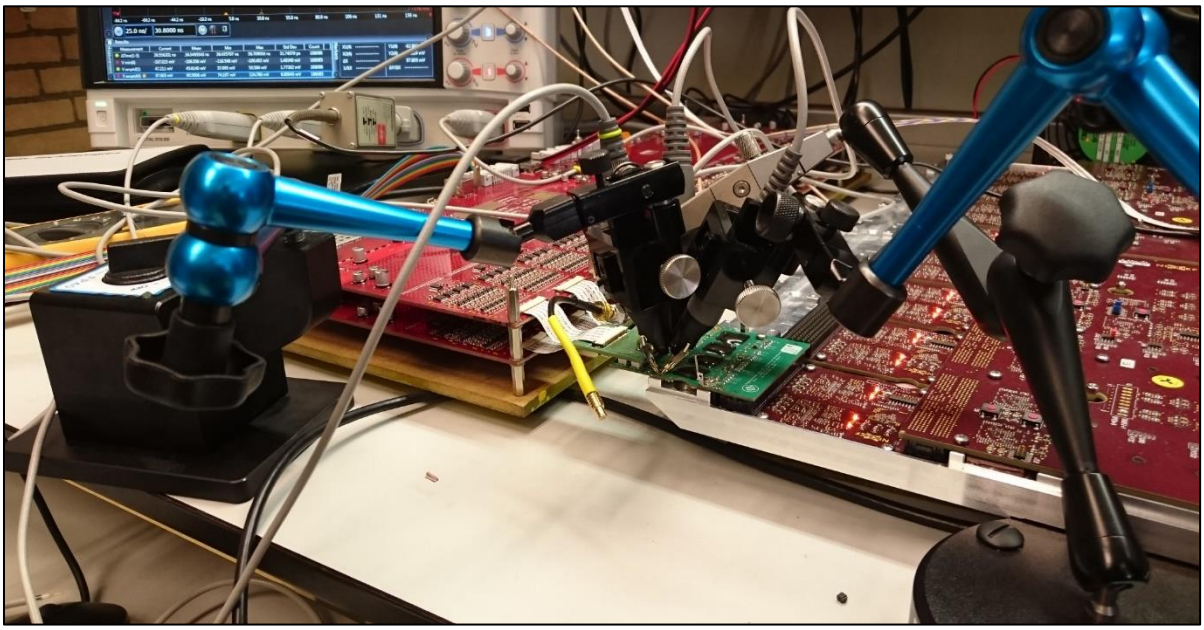


Figura 84 – Pontas de prova conectadas às portas de *DEBUG* do **PACIFIC**.

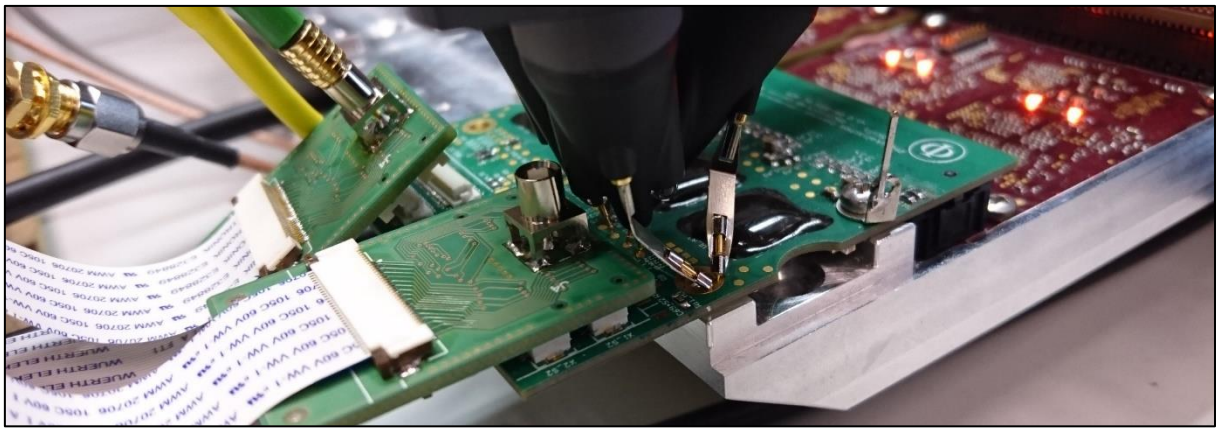


Figura 85 – Detalhe da conexão das pontas de prova ativas.

4.1.2. Análise da Saída do Injetor

Começou-se esta sequência de testes com a caracterização do Módulo Injetor através de formas de onda do pulso de saída do canal 8, no exemplar S/N #1. A referida saída foi desconectada do chip **PACIFIC** e conectada diretamente ao osciloscópio. Uma varredura de amplitude e largura foi então executada. Esta varredura consistiu na variação da configuração de amplitude do pulso ($aDAC$) em 16 níveis, diminuindo-se seu valor $17\ LSB$ a cada nível, de $0xFF$ até $0x00$, para determinada configuração de largura ($wDAC$) do pulso. Em seguida, a configuração de largura foi aumentada em $1\ LSB$, e a varredura de amplitude foi repetida. Este procedimento foi executado quatro vezes, para $wDAC = 0x02$, $0x03$, $0x04$ e $0x05$. As 64 formas de onda vistas no Apêndice F apresentam as medidas resultantes do processamento estatístico em sua parte inferior. A partir destas medidas processou-se a análise dos resultados.

Um gráfico com as medidas de área do pulso, calculada na forma da integral dos valores de tensão exibidos na tela do osciloscópio, em função de $aDAC$ para vários valores de $wDAC$ é visto na Figura 86. A seguir, determinou-se através da relação abaixo a carga do pulso em função de sua área:

$$Q_{pulso}@\ 50\Omega\ [C] = {Area_{pulso}[Vs]}/{50[\Omega]} \tag{9}$$

Um gráfico com a carga estimada do pulso em função de $aDAC$ para vários valores de $wDAC$ é visto na Figura 87. Medidas da tensão de pico do pulso também foram registradas e são mostradas no gráfico da Figura 88.

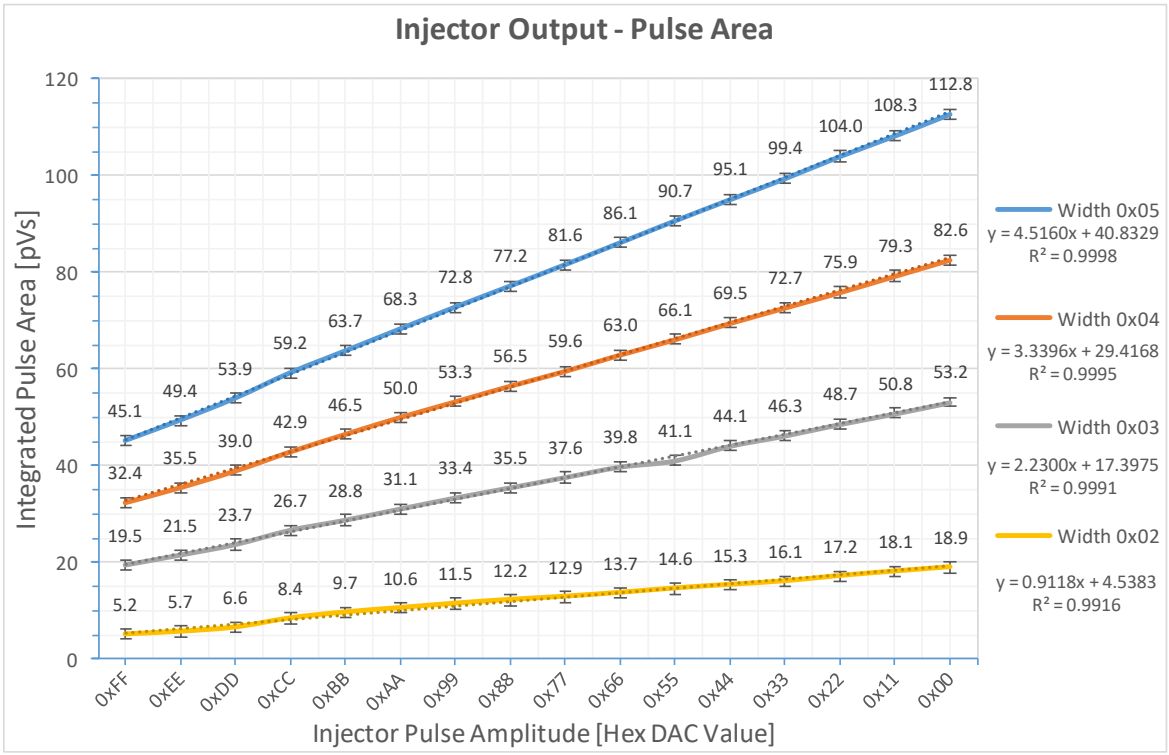


Figura 86 – Área do pulso em função de $aDAC$, para diversos valores de $wDAC$.

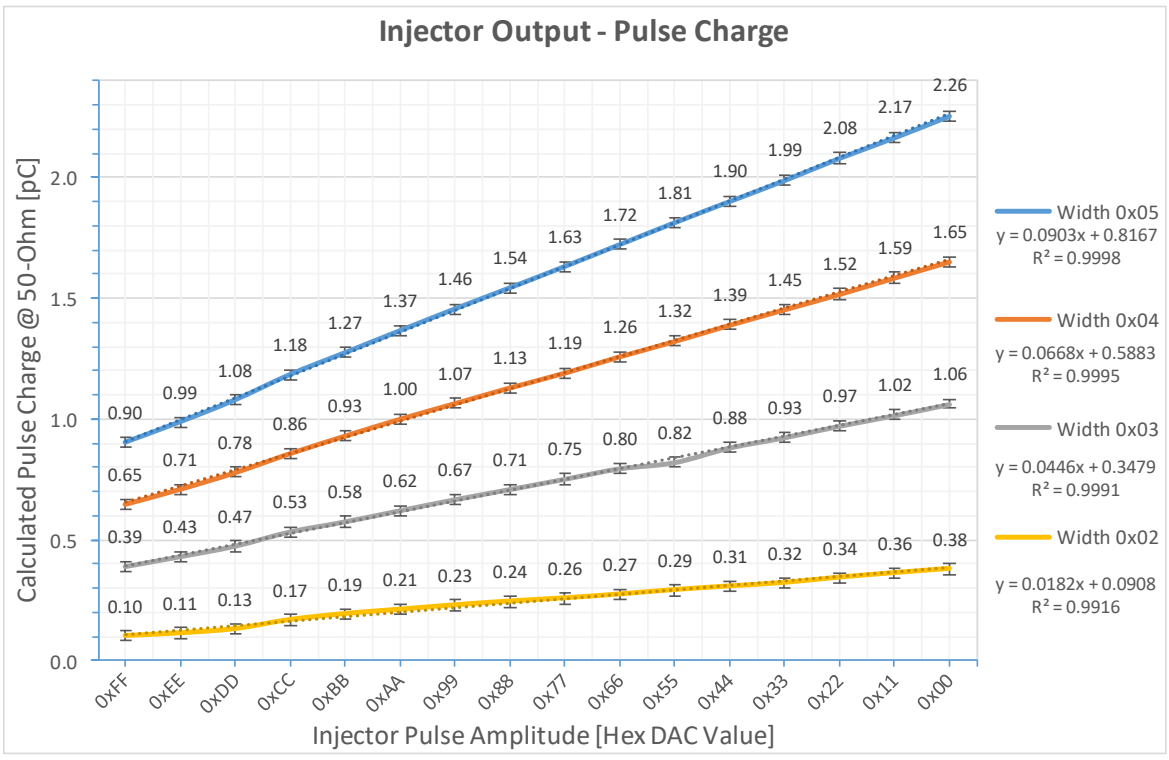


Figura 87 – Carga de saída em função de $aDAC$, para diversos valores de $wDAC$.

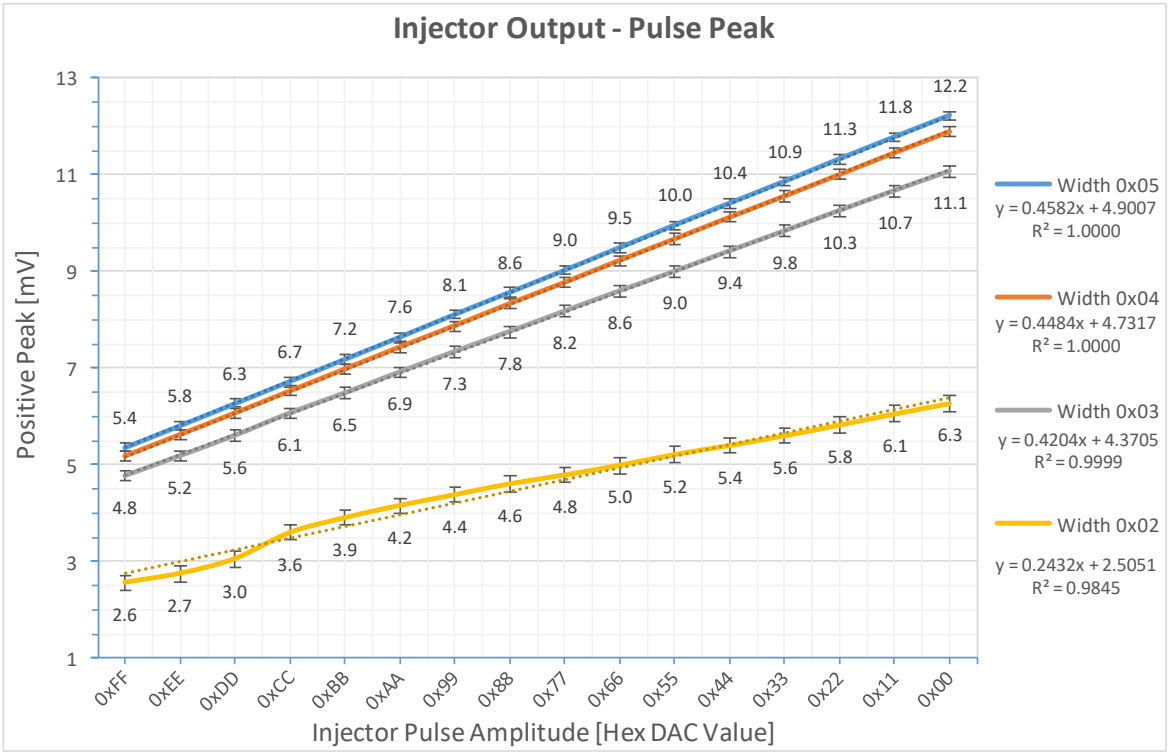


Figura 88 –Tensão de pico do pulso em função de $aDAC$, para vários $wDAC$.

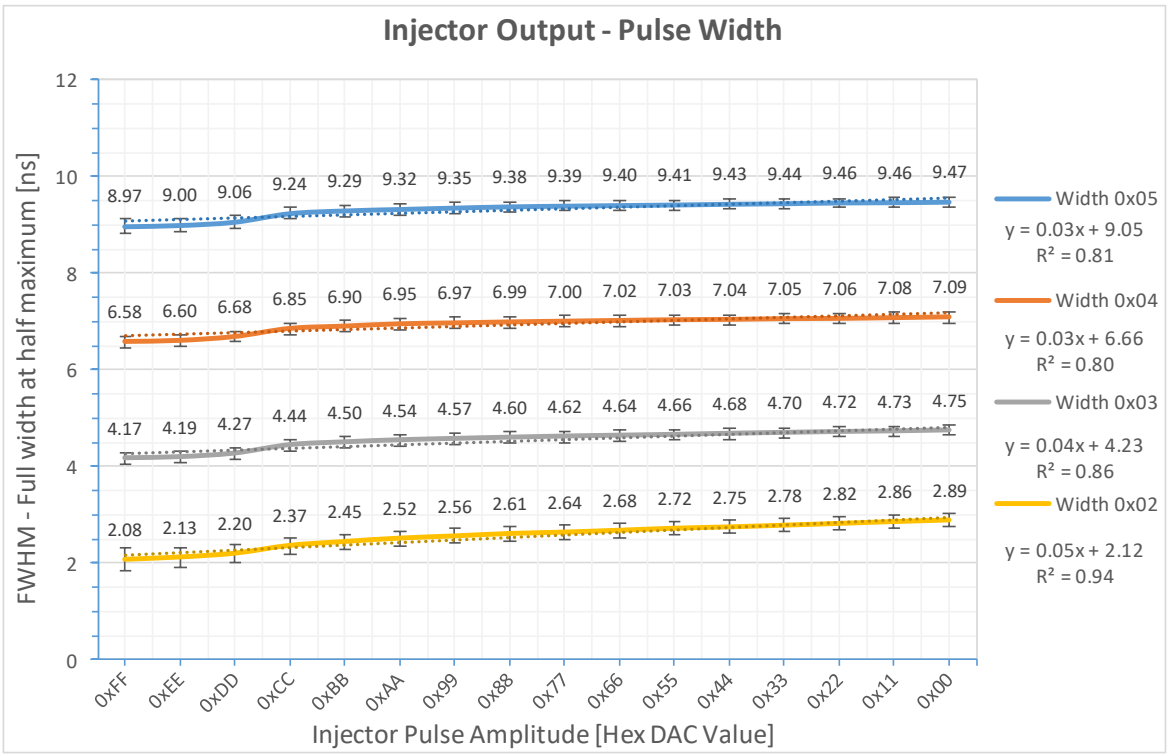


Figura 89 – Largura do pulso em função de $aDAC$, para vários $wDAC$.

Observou-se que o formato trapezoidal do pulso causa pequena deriva em sua largura efetiva com a variação de $aDAC$, mesmo para um $wDAC$ constante. Este efeito foi estudado e encontra-se demonstrado no gráfico da Figura 89, onde se observa a

largura do pulso à meia altura (FWHM) como função de $aDAC$ para diferentes valores de $wDAC$. Nota-se que este efeito é mais intenso em pulsos de menor área.

Esta avaliação de um dos canais de saída do Módulo Injetor permitiu uma primeira avaliação qualitativa da resposta de diversos parâmetros elétricos a valores programados nos registradores $aDAC$ e $wDAC$. A partir das medições realizadas foi possível montar a Tabela 1, abaixo, que será de grande importância para a análise a seguir.

Tabela 1 – Carga do pulso em função de diferentes parâmetros de amplitude e largura.

	Calculated Charge @ 50 Ohm [pC]							
DAC	Width 0x02	Std. Dev.	Width 0x03	Std. Dev.	Width 0x04	Std. Dev.	Width 0x05	Std. Dev.
0xFF	0.10307	0.02140	0.38921	0.02114	0.64775	0.01975	0.90228	0.02065
0xEE	0.11402	0.02088	0.43050	0.02095	0.71008	0.02000	0.98700	0.02058
0xDD	0.13152	0.02146	0.47417	0.02085	0.77902	0.02029	1.07825	0.02023
0xCC	0.16893	0.02107	0.53344	0.02094	0.85813	0.02037	1.18339	0.02052
0xBB	0.19341	0.02108	0.57537	0.02092	0.93076	0.02021	1.27420	0.02060
0xAA	0.21127	0.02146	0.62180	0.02109	1.00076	0.02023	1.36545	0.02040
0x99	0.22936	0.02185	0.66741	0.02079	1.06539	0.02035	1.45580	0.02054
0x88	0.24492	0.02210	0.70977	0.02097	1.13042	0.02066	1.54337	0.02058
0x77	0.25808	0.02158	0.75246	0.02134	1.19181	0.02068	1.63196	0.02084
0x66	0.27390	0.02162	0.79596	0.02047	1.25948	0.02041	1.72278	0.02116
0x55	0.29195	0.02194	0.82170	0.02054	1.32278	0.02108	1.81358	0.02059
0x44	0.30680	0.02168	0.88253	0.02019	1.39018	0.02096	1.90150	0.02090
0x33	0.32144	0.02203	0.92518	0.02038	1.45416	0.02063	1.98846	0.02079
0x22	0.34312	0.02207	0.97304	0.02059	1.51809	0.02111	2.08058	0.02073
0x11	0.36154	0.02220	1.01674	0.02016	1.58559	0.02102	2.16570	0.02127
0x00	0.37898	0.02288	1.06364	0.02029	1.65263	0.02106	2.25576	0.02091

Dados tabulados para os demais parâmetros elétricos de pulso do Módulo Injetor obtidos nesta sequência de testes estão disponíveis para consulta no Apêndice G.

4.1.3. Análise da Resposta do *ASIC* PACIFIC

A resposta do *ASIC* **PACIFIC *4b*** sob excitação do Módulo Injetor foi estudada monitorando suas saídas *DEBUG*. Uma indicação destas saídas analógicas de *DEBUG* foi fornecida na Figura 30. A presente análise baseou-se no processamento das amplitudes das saídas *DEBUG_PRE* (sinal logo após o estágio de pré-amplificação) e *DEBUG_TH* (sinal de saída do *Track & Hold*, entre integração e digitalização). Devido

à notável presença de ruído, justificado em parte pelo uso das pontas de alta impedância, optou-se pelo cálculo da amplitude a partir das seguintes medições:

- *Baseline* $[mV]$ e *Positive Peak* $[mV]$ do sinal *DEBUG_ PRE*;
- *Baseline* $[mV]$ e *Negative Peak* $[mV]$ do sinal *DEBUG_ TH*.

De forma que:

$$DEBUG_PRE_{Amplitude} = Positive\ Peak_PRE_{Mean} - Baseline_PRE_{Mean} \quad (10)$$

$$DEBUG_TH_{Amplitude} = Baseline_TH_{Mean} - Negative\ Peak_TH_{Mean} \quad (11)$$

A Figura 90, abaixo, ilustra cada uma dessas medidas sobre o sinal:

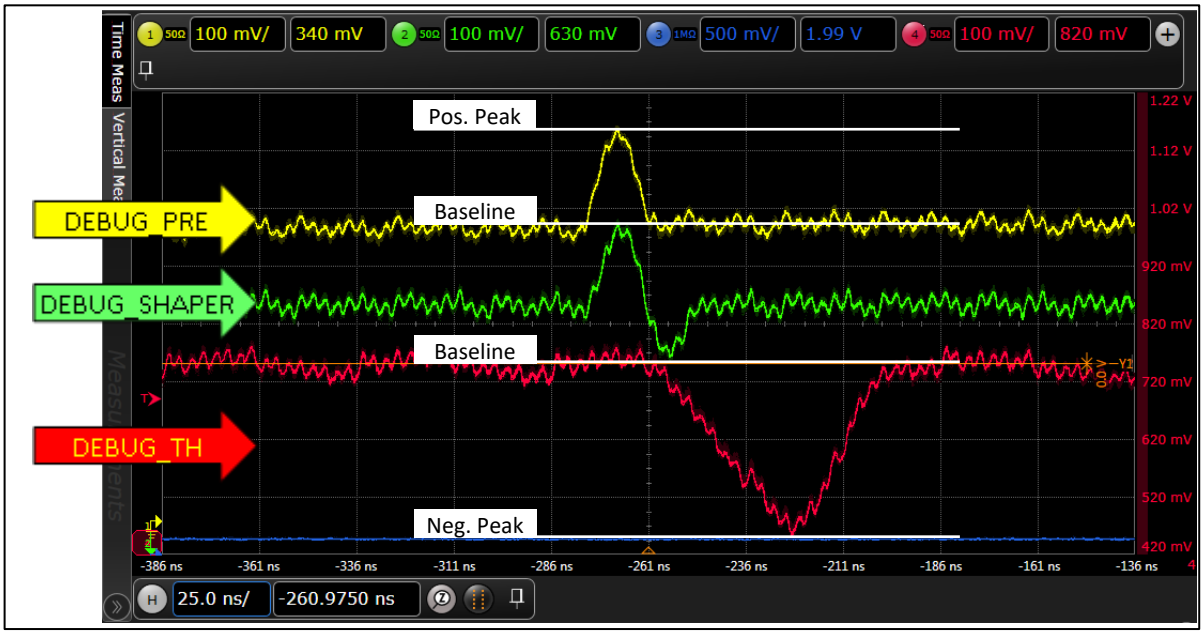


Figura 90 – Tela de osciloscópio com os sinais *PACIFIC DEBUG*, que ilustra como foram obtidas as amplitudes destes sinais.

4.1.3.1. Resposta do Estágio Integrador

A resposta do integrador (*Track & Hold*) é mostrada nos gráficos da Figura 91 em função de *aDAC* (amplitude do pulso injetado) para diferentes valores de *wDAC* (largura do pulso injetado) e de ganhos de entrada no **PACIFIC**. Estes mesmos dados foram combinados no gráfico da Figura 92.

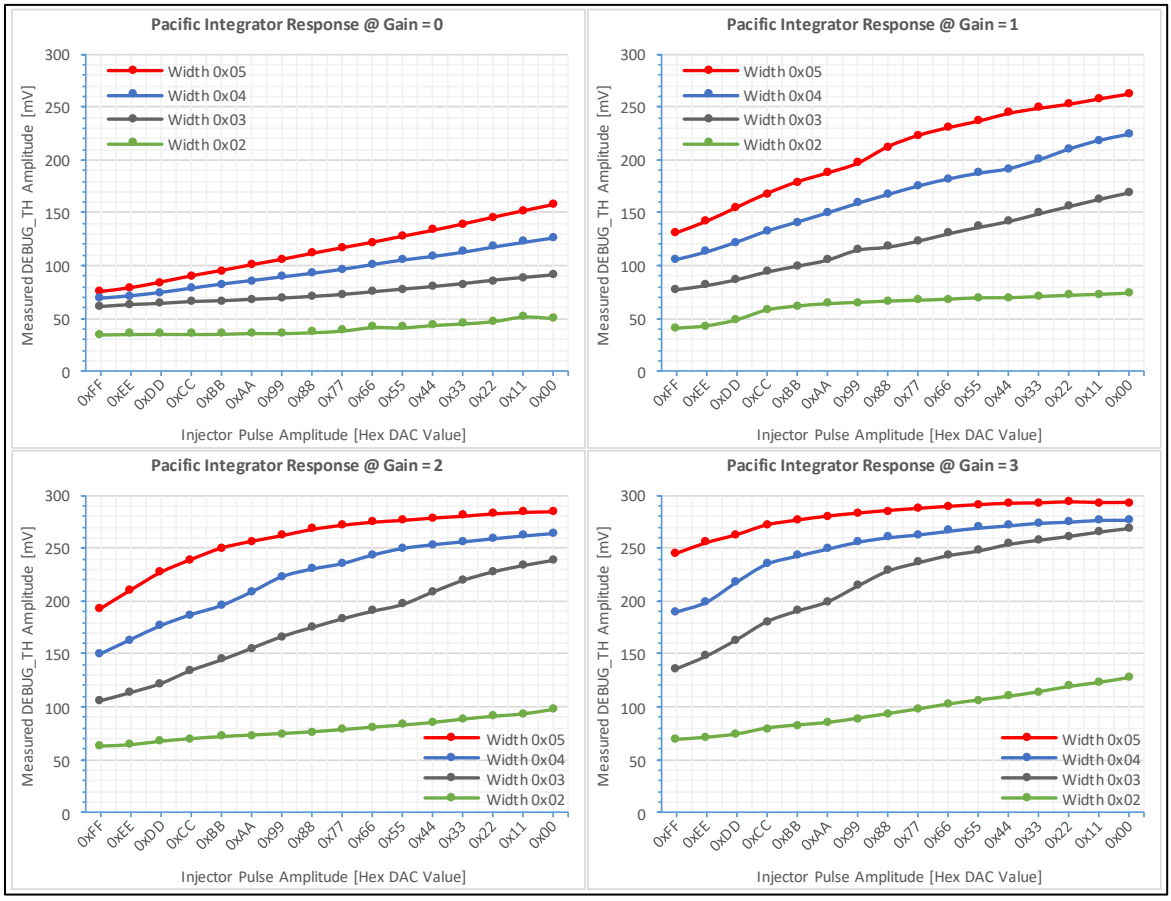


Figura 91 – Amplitude do *DEBUG_ TH* em função da amplitude do pulso do injetor para quatro larguras de pulso. Os quatro painéis mostram os resultados para os quatro ganhos do **PACIFIC**: Baixo (superior esquerdo); Médio (superior direito); Médio-Alto (inferior esquerdo); Alto (inferior direito).

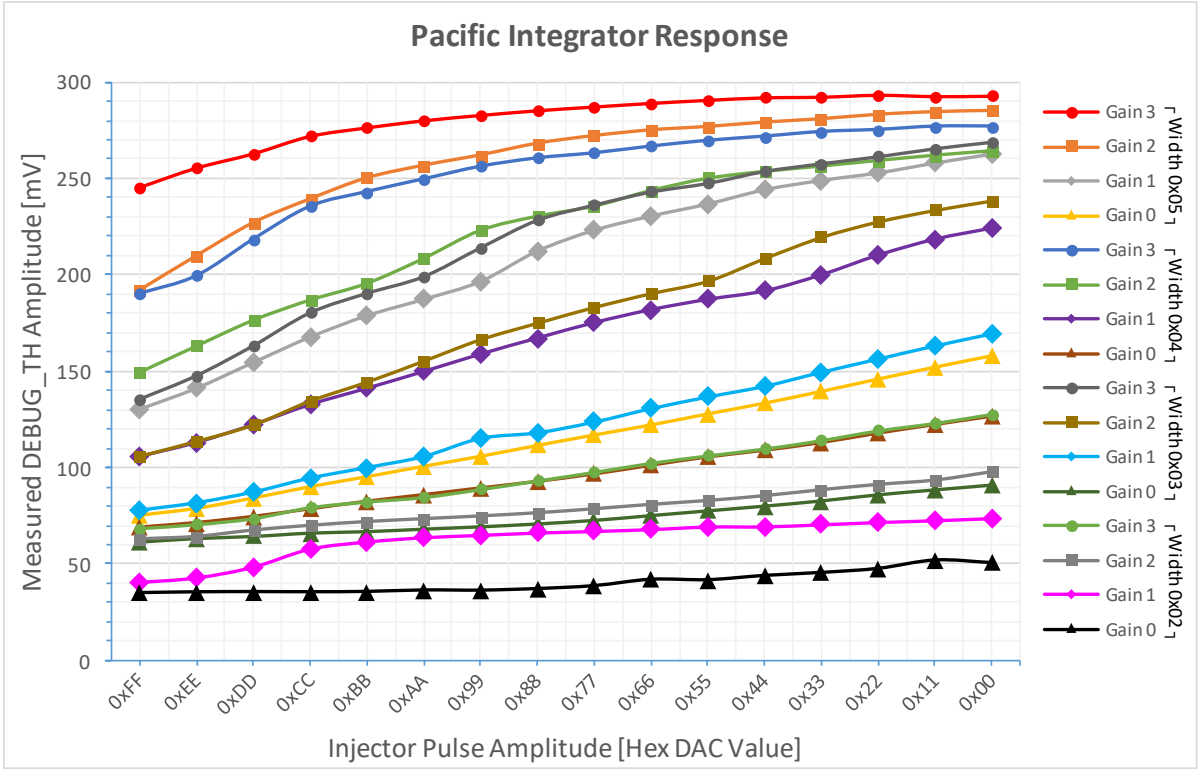


Figura 92 – Comportamento do sinal *DEBUG_ TH* em função da amplitude do pulso do injetor para quatro diferentes larguras de pulso e quatro diferentes ganhos.

As mesmas medidas são ainda exibidas no gráfico da Figura 93 como função da carga de injeção, estimada a partir da Tabela 1. Linhas tracejadas neste gráfico indicam ajustes lineares calculados para cada configuração de ganho do **PACIFIC**. Não foram consideradas nos ajustes as medidas circuladas em vermelho. A inclinação das quatro retas ajustadas está aproximadamente na razão [0,5 : 1,0 : 1,5 : 2,0] esperada para os ganhos nominais do **PACIFIC**, o que indica uma boa resposta dinâmica.

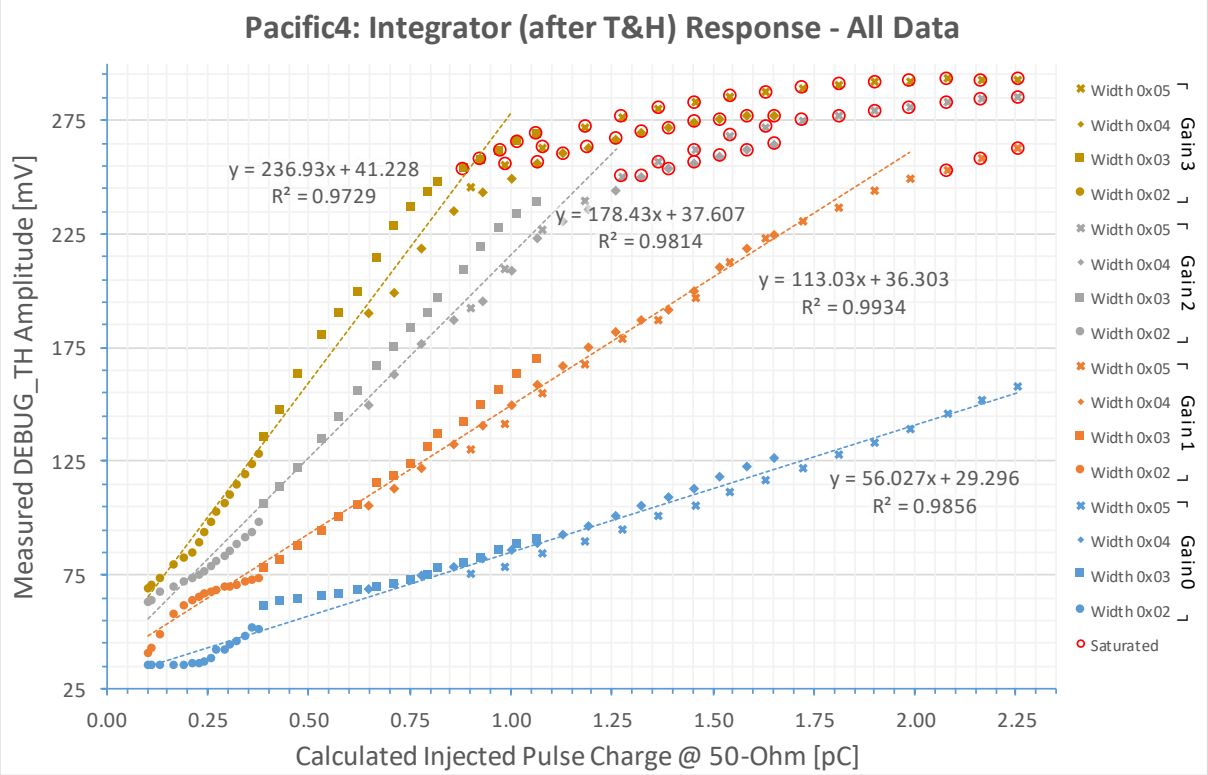


Figura 93 – Amplitude de *DEBUG_ TH* em função da carga injetada estimada para quatro diferentes larguras de pulso do injetor e quatro diferentes ganhos do **PACIFIC**.

Nos últimos gráficos nota-se que ocorre saturação do sinal *DEBUG_ TH*. Se isso for devido a uma saturação real do estágio de integração do **PACIFIC**, então fica demonstrado que o Módulo Injetor foi capaz de varrer todo seu intervalo dinâmico. Entretanto, como demonstrado na Figura 94, notou-se que o sinal *DEBUG_ TH* apresenta uma rampa de descida que limita o deslocamento durante o período de uma janela de integração, o que coloca em dúvida esta interpretação. Posteriormente, descobriu-se que uma modificação no *design* do *chip* **PACIFIC** limitou a potência dos *buffers* que processam o sinal *DEBUG*, comprometendo seu *slew rate*.

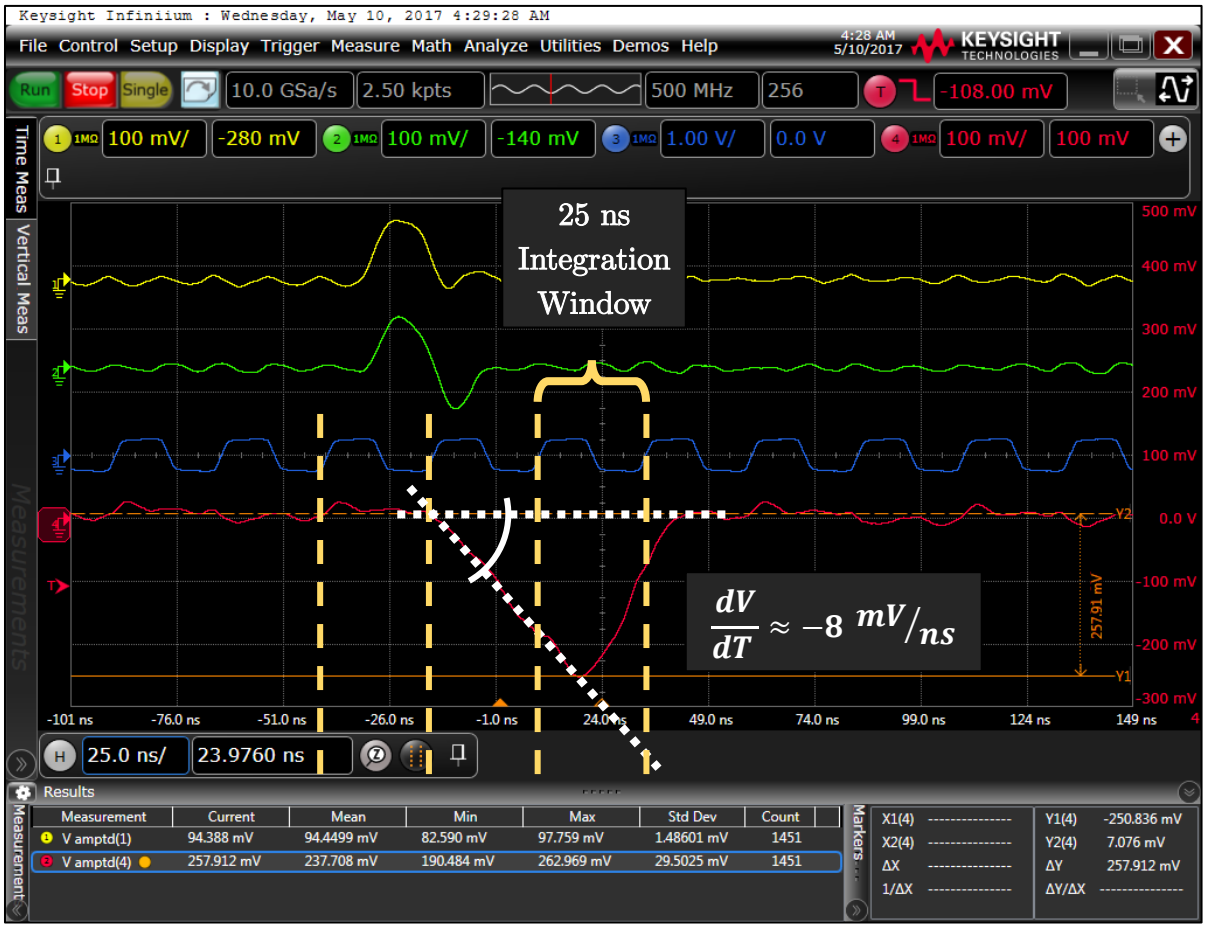


Figura 94 – Detalhe da rampa presente no sinal *DEBUG_TH*.

4.1.3.2. Resposta do Estágio de Pré-Amplificação

Objetivando isolar o estágio causador da saturação observada na Figura 93 procedeu-se também à análise da resposta do estágio pré-amplificador.

A amplitude do sinal de saída deste estágio para diferentes ganhos do **PACIFIC** e valores de $wDAC$ é mostrada nos gráficos da Figura 95 em função da tensão de pico do pulso injetado. Nota-se uma distribuição praticamente linear ($R^2 \approx 1$) destas medidas, exceto para $FWMH \approx 2{,}6\,ns$. Justifica-se esta exceção pela quantidade muito pequena de carga para pulsos tão rápidos, que colocam o estágio de pré-amplificação para trabalhar muito próximo ao ruído de fundo. Estes dados são mesclados na Figura 96. Mais uma vez, nota-se inclinações das quatro retas ajustadas bastante próximas à razão $[0{,}5 : 1{,}0 : 1{,}5 : 2{,}0]$ esperada para os ganhos nominais do **PACIFIC**.

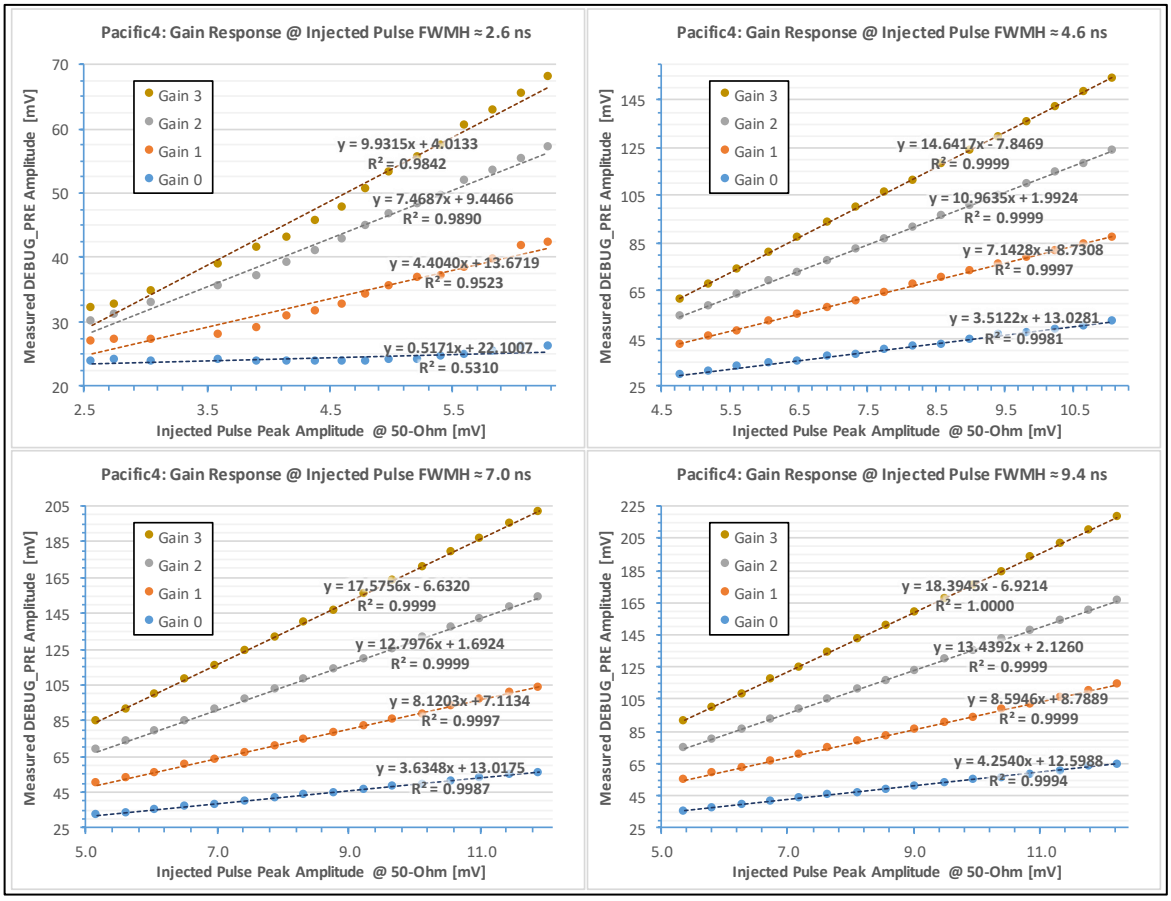


Figura 95 – Sinal *DEBUG_PRE* em função da amplitude do pulso injetado para quatro ganhos (L, M, MH, H). Os quatro painéis mostram os resultados para quatro diferentes larguras de pulso do injetor: 0x02 (superior esquerdo); 0x03 (superior direito); 0x04 (inferior esquerdo); 0x05 (inferior direito).

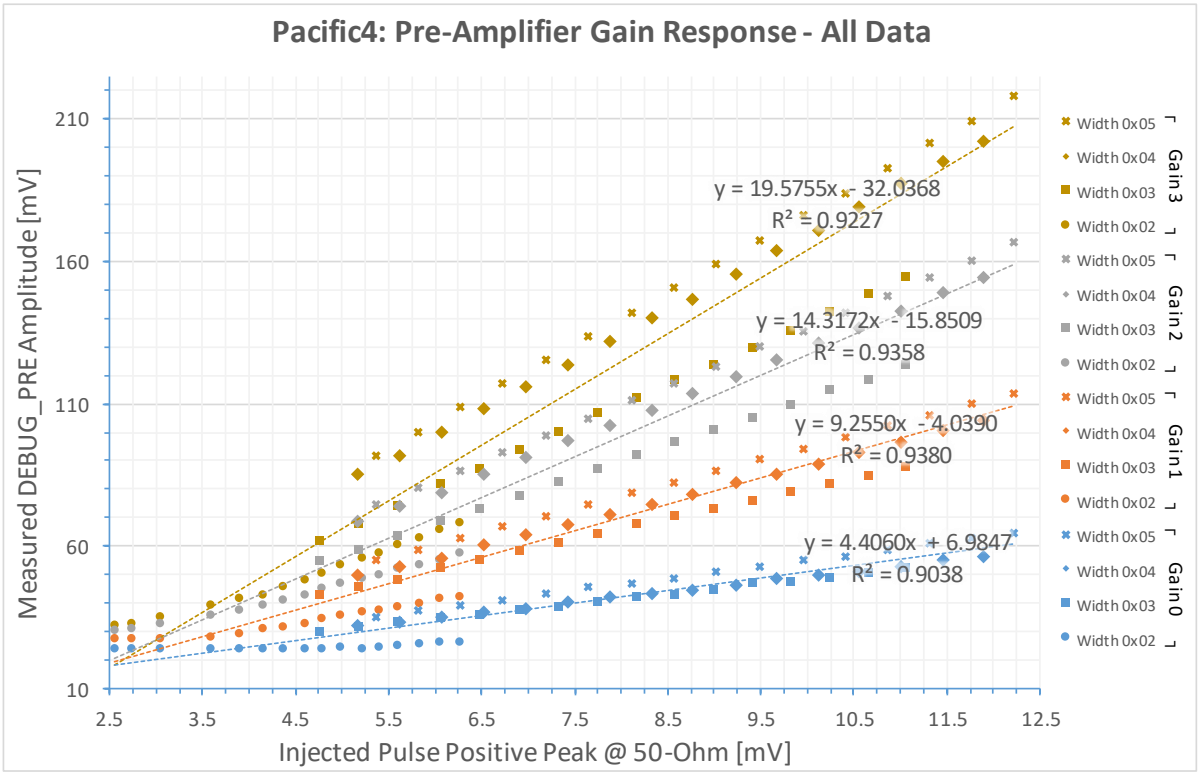


Figura 96 – Amplitude do *DEBUG_PRE* em função da amplitude do pulso injetado em quatro diferentes larguras de pulso e para quatro diferentes ganhos do **PACIFIC**.

4.2. Segunda Sequência de Testes

Embora a primeira sequência de testes tenha fornecido importantes dados de calibração do Módulo Injetor, é importante notar que foram processadas medições de apenas um dos 256 canais de saída. Notou-se durante a interpretação dos resultados que a estatística obtida é insuficiente para generalizar sua calibração. Realizou-se então em dezembro de 2017, no CBPF, com o exemplar Número de Série #2 do Módulo Injetor uma segunda sequência de testes com o propósito de estabelecer parâmetros gerais de calibração mais robustos.

Nesta sequência foram analisados 64 canais de saída, 1/4 do total, registrando-se com o osciloscópio em cada um deles, para as 112 combinações de $aDAC$ e $wDAC$ testadas, as seguintes medidas: Área, *Timing*, Largura, Tensão de Pico e *Baseline*.

O *ASIC* **PACIFIC** não foi analisado neste momento pois ainda se aguarda o lançamento da versão ***5***, a qual deverá corrigir o problema com os pinos *DEBUG*.

4.2.1. Ferramentas e Metodologia

Para uma medição precisa de parâmetros elétricos dos pulsos gerados pelo Módulo Injetor utilizou-se o osciloscópio *Agilent Infiniium* modelo *DSO-90254A,* S/N MY48240306, localizado na sala de eletrônica do LAFEX. Este instrumento possui rotinas de auto calibração que são executadas com certa periodicidade, e foi considerado como padrão de referência para as medições da segunda sequência de testes.

Todas estas medidas foram obtidas adquirindo-se o sinal das saídas do Módulo Injetor através de acoplamento direto com o osciloscópio (utilizando sua terminação interna de $50\,\Omega$). Para isto desenvolveu-se uma placa adaptadora especial. A bancada de testes utilizada nesta sequência é vista na Figura 97. A conexão do Módulo Injetor com o osciloscópio é vista em detalhes na Figura 98.

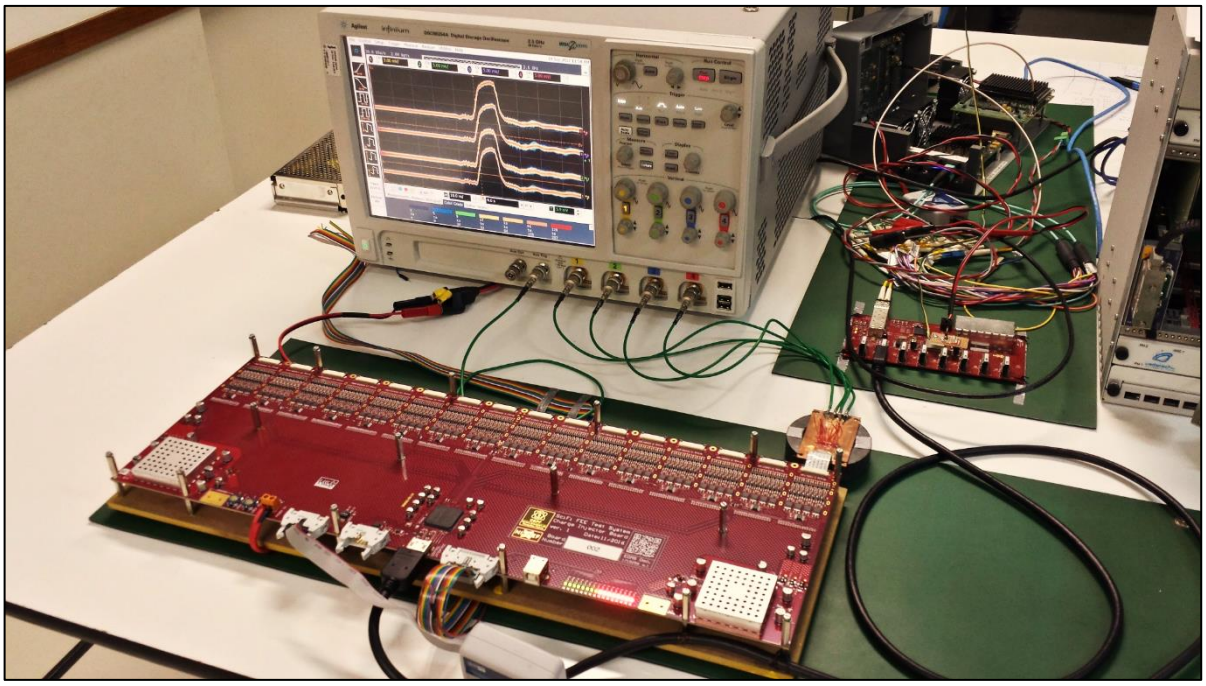


Figura 97 – Bancada da segunda sequência de testes.

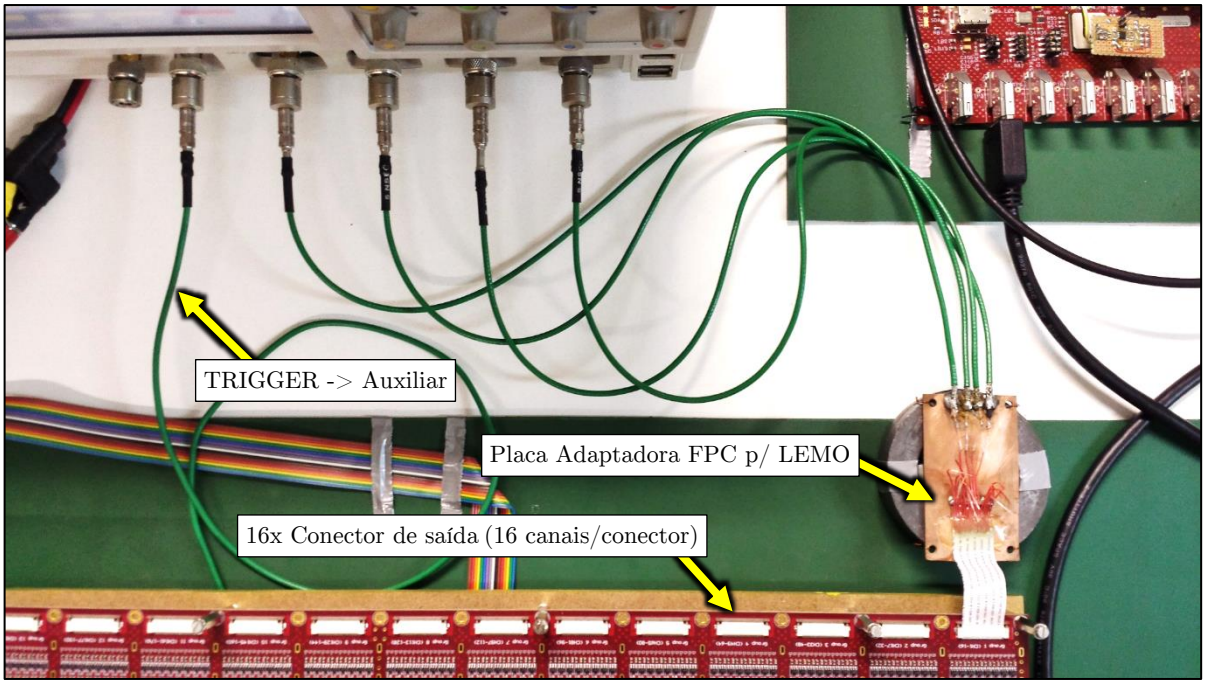


Figura 98 – Detalhe da conexão do Módulo Injetor no osciloscópio.

Um cabo LEMO adicional foi conectado a uma das saídas SPARE do Módulo Injetor, programada em FIRMWARE para fornecer uma cópia atrasada do sinal de disparo de $400\,kHz$, que por sua vez é derivado da base de tempo local fornecida pelo cristal interno de $40\,MHz$ do Módulo Injetor. Este sinal foi utilizado como fonte de *trigger* do osciloscópio e serviu como referência global de *timing*.

4.2.2. Automação da Aquisição de Dados

A sequência de testes proposta requer a aquisição de 16 formas de onda por largura por canal, isto equivale a um total de $16 \times 7 \times 64 = 7.168$ formas de onda. Esta escala inviabilizou a tomada de dados manual praticada durante a primeira sequência e motivou o desenvolvimento de uma ferramenta para comando sincronizado e controle centralizado do Módulo Injetor e do Osciloscópio.

O Módulo Injetor foi desenhado para controle remoto, podendo receber comandos pelo barramento I^2C e operar em modo *standalone*. Desta forma, uma solução para controle do Módulo Injetor precisa levar em conta apenas a implementação de uma interface *Master* I^2C padrão.

O fabricante do osciloscópio *Agilent Infiniium* disponibiliza diversos meios para seu controle remoto. Entretanto, os meios convencionais exigem a compilação de um *software* exclusivo para a finalidade e obrigam a utilização de um microcomputador para a execução deste *software*. A simplicidade dos comandos necessários a esta aplicação específica, e a possibilidade de centralizar todo o algoritmo em um único dispositivo (consequentemente, em um único código) convidou a um olhar diferente.

A plataforma *open-source* de prototipagem rápida *Arduino Due* possui um controlador *USB* nativo implementado, permitindo a emulação de um teclado *USB* físico e o pressionamento da sequência apropriada de teclas através do programa gravado no microcontrolador. Por sua vez, a interface de usuário do osciloscópio dispõe de uma série de atalhos de teclado, o que fez com que este método fosse considerado. Uma vez testada, esta abordagem demonstrou excelentes resultados.

Para os testes propostos, o Módulo Injetor foi programado para injetar continuamente ($400kHz$) uma série de pulsos idênticos no canal selecionado do osciloscópio, que está configurado para processar diversas medidas estatísticas desta série de pulsos. Transcorrido o tempo de aquisição estipulado em 15 segundos aproximadamente mil

formas de onda terão sido processadas e o osciloscópio deve ser comandado para parar a aquisição, gravar em disco as medidas selecionadas na forma de um arquivo **`*.csv`**, gravar uma captura de tela na forma de um arquivo **`*.png`** e fica em espera. Em seguida, o Módulo Injetor deve ser reprogramado com os novos parâmetros de pulso. O osciloscópio então deve reiniciar a aquisição pelo tempo necessário até que, novamente, seja atingida a estatística desejada. Desta forma o ciclo se completa e reinicia. O algoritmo da Figura 99 ilustra este método.

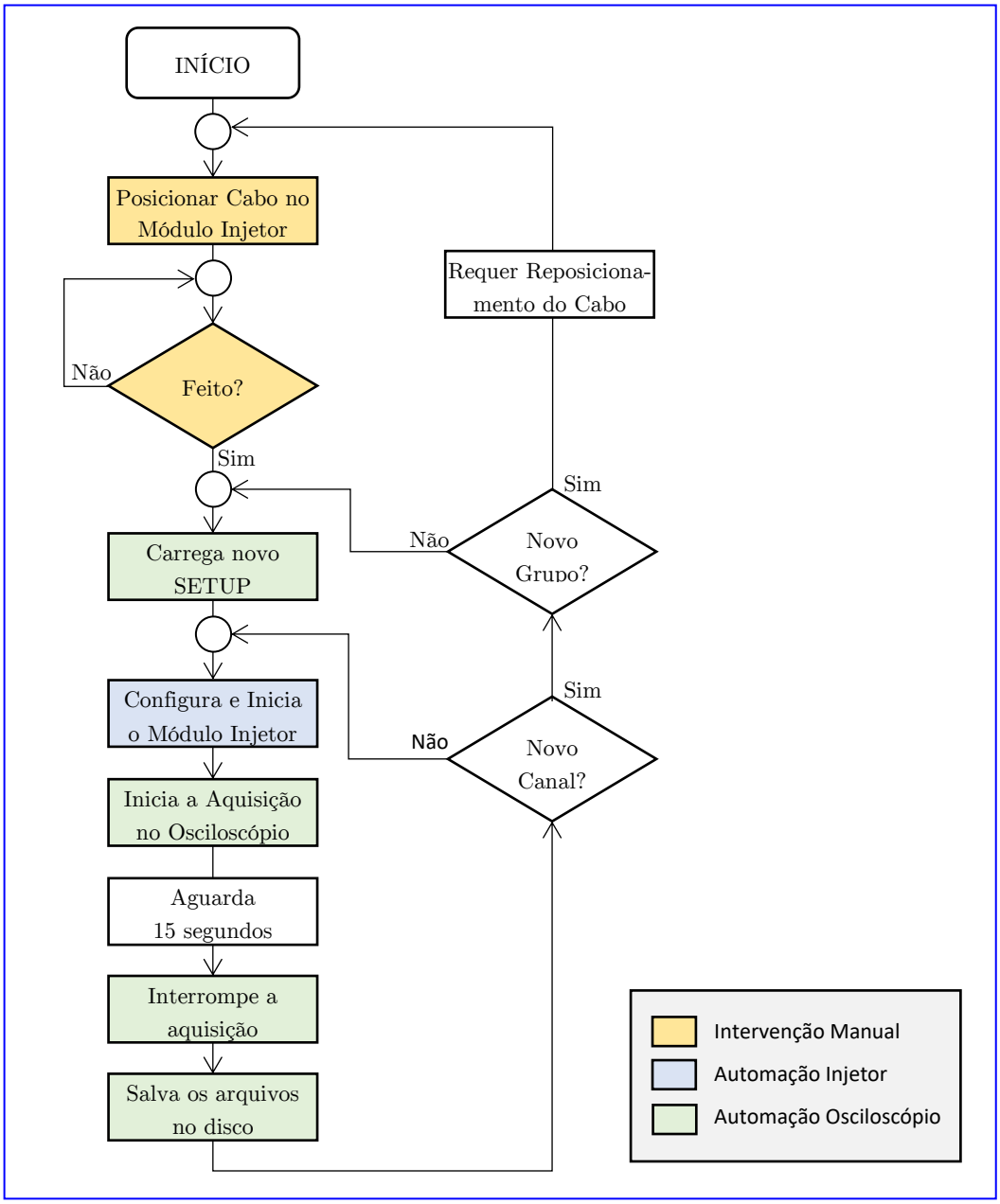


Figura 99 – Algoritmo de teste do Módulo Injetor.

O *hardware* Arduino DUE foi escolhido como elemento central de implementação deste algoritmo de teste pois possui duas portas *USB*, de modo que enquanto uma está conectada ao osciloscópio e é utilizada para emulação do teclado a outra pode ser conectada a um microcomputador para visualização de mensagens e execução de comandos de usuário em terminal serial emulado. O dispositivo possui ainda a vantagem adicional de operar em 3v3, uma das tensões de I/O do Módulo Injetor, dispensando assim o uso de um *level shifter* I^2C. O diagrama da Figura 100 ilustra a implementação em *hardware* desta solução.

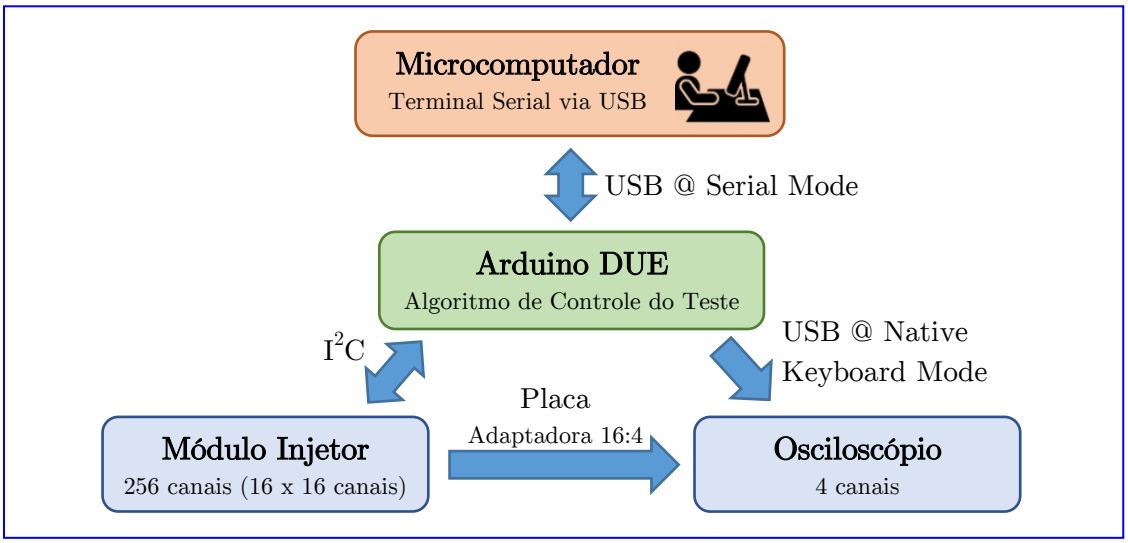


Figura 100 – Diagrama em blocos da implementação do algoritmo de teste.

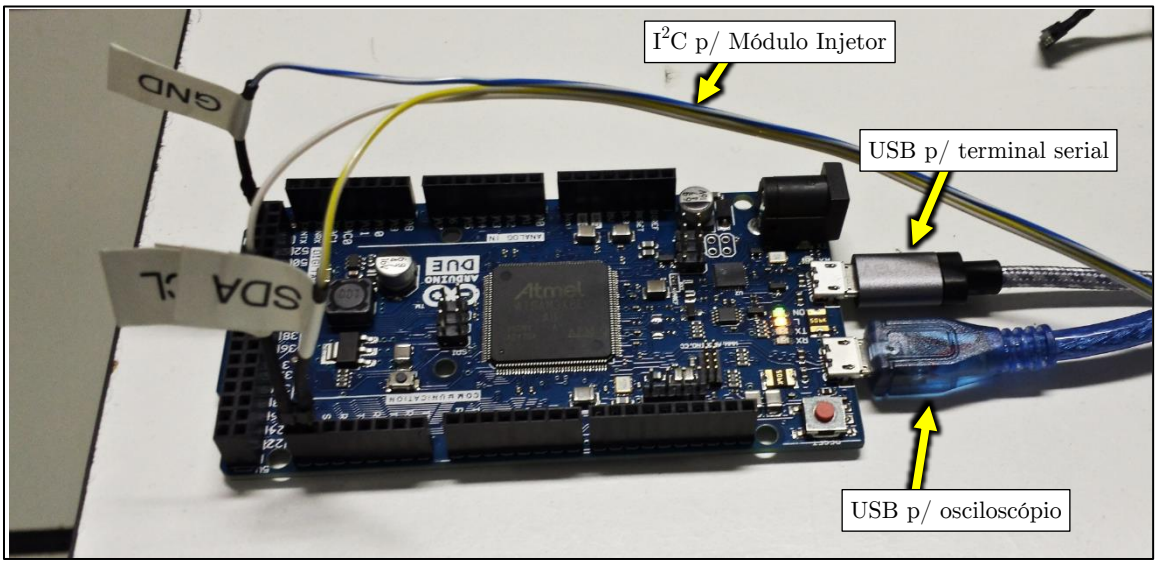


Figura 101 – *Hardware* utilizado para a implementação do algoritmo de teste.

Na Figura 101, acima, nota-se a simplicidade do *hardware* utilizado para a implementação do algoritmo de teste que consiste, basicamente, apenas no *Arduino* DUE e nos cabos de interconexão. Na Figura 102, abaixo, é possível ver uma captura de tela da interface de usuário emulada como terminal serial em uma das portas *USB*.

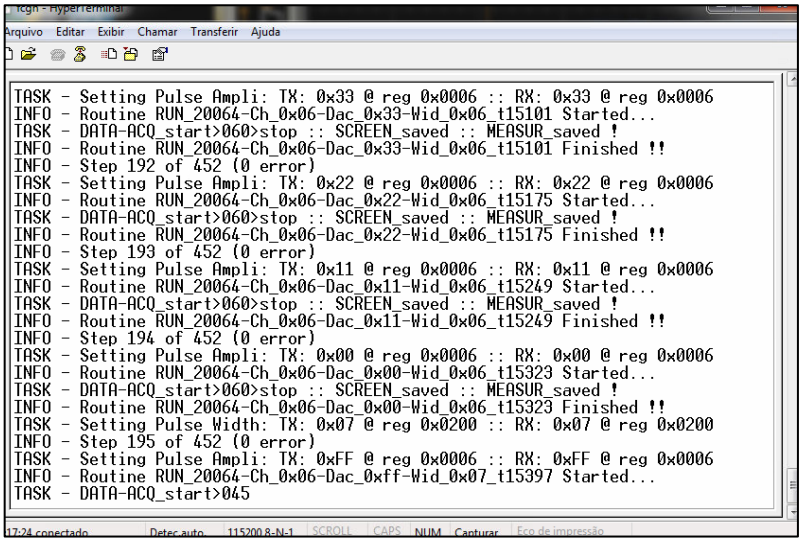


Figura 102 – Captura de tela da interface de usuário do algoritmo de teste.

Após a execução completa do algoritmo obtêm-se um arquivo **`*.csv`** e um arquivo **`*.png`** para cada configuração testada no Módulo Injetor. Estes arquivos foram separados em diretórios correspondentes ao grupo de canais ao qual a medida pertence, como visto na Figura 103. Uma amostra da captura de tela pode ser vista na Figura 104 e uma amostra do arquivo de medidas na Figura 105.

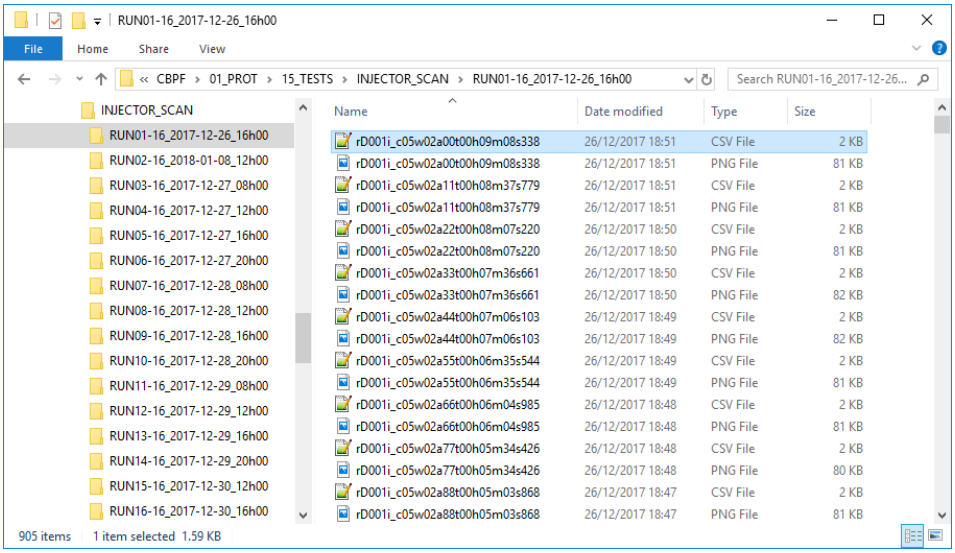


Figura 103 – Diretório com os arquivos produzidos pelo osciloscópio.

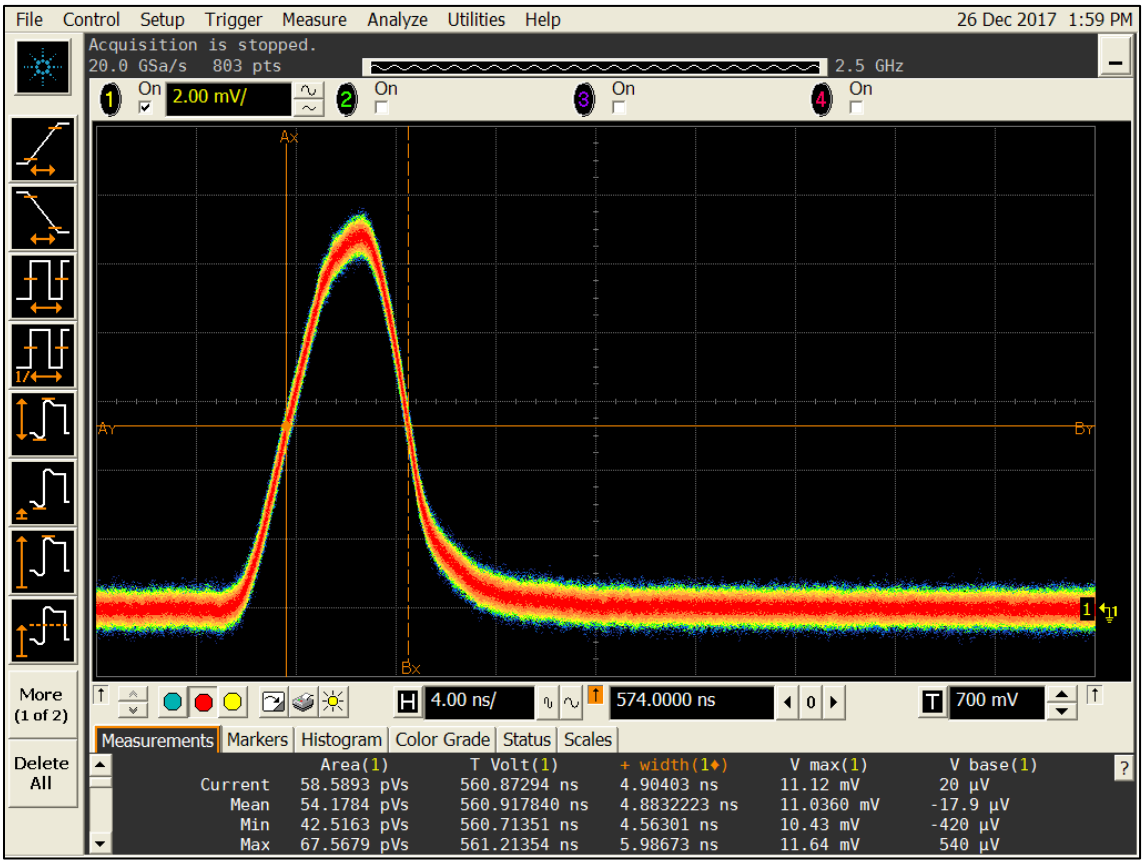


Figura 104 – Exemplo de arquivo salvo em disco com a captura de tela.

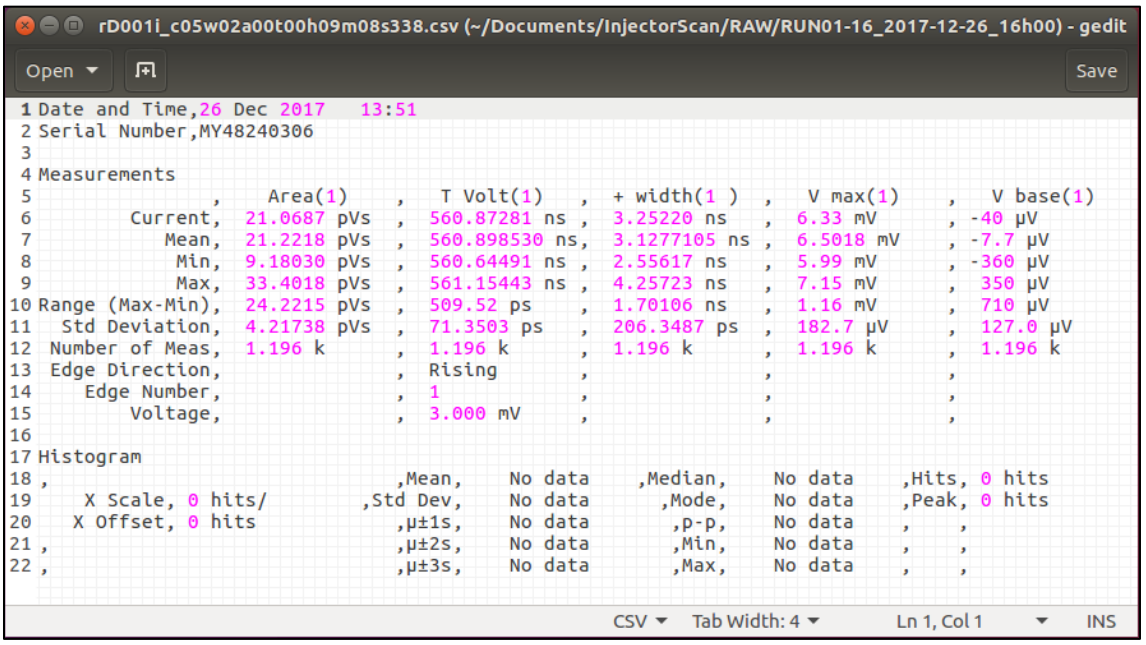


Figura 105 – Exemplo de arquivo salvo em disco com o resultado das medições.

Para cada medida ficam registrados o valor médio, o mínimo, o máximo, o desvio padrão e ainda outros parâmetros complementares.

4.2.3. Processamento dos Dados

O conjunto de dados brutos produzido pelo osciloscópio encontra-se fragmentado em $16 \times 7 \times 64 = 7.168$ arquivos, um para cada configuração de pulso por canal. A análise deste conjunto de dados requer sua desfragmentação e construção de uma matriz única. Para isto, foi desenvolvido o *script* em **PYTHON** reproduzido no Apêndice I, capaz de varrer toda a árvore de subdiretórios indicada buscando pelos arquivos `*.csv`, reorganizando os dados e os juntando em um único arquivo `*.csv` de saída.

A partir deste arquivo único utilizou-se outro *script* em **PYTHON**, reproduzido no Apêndice J, para executar as análises e obter os resultados que serão apresentados a seguir. O *Framework* **ROOT** desenvolvido pelo CERN [42] foi utilizado para ajustes de funções e geração de gráficos, histogramas e tabelas. O fluxograma da Figura 106 ilustra como os dados brutos foram tratados e analisados.

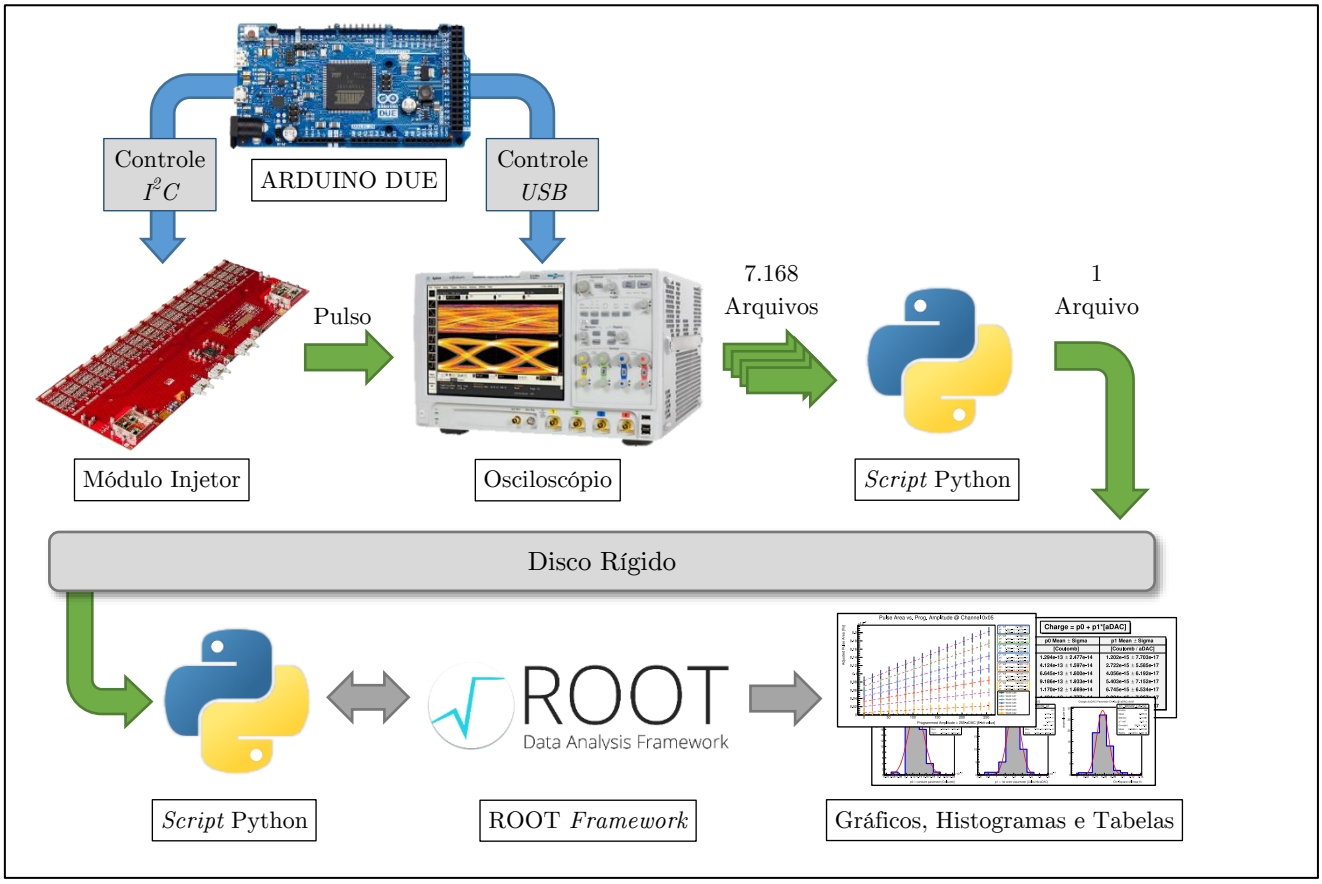


Figura 106 – Fluxograma de tratamento e análise dos dados.

4.2.3.1. Correção da Medida de Área

Durante a tomada de dados preliminar, notou-se que a escala $2\ mV/div$ utilizada cria uma condição na qual há, proporcionalmente, grande variação na medida da área para pequenas variações no *baseline* (nível DC) do sinal. Ao longo dos vários dias de testes diversas condições (como variações na temperatura da sala e presença de ruídos ocasionais) afetam o *baseline* do sinal adquirido, como visto na Figura 107.

A documentação do osciloscópio especifica que o nível DC da entrada deve ser periodicamente ajustado para compensar este efeito corrigindo, assim, o *baseline* de cada canal. Tal ajuste tem de ser refeito toda vez que as conexões se alteraram. Dadas a longa duração da tomada de dados e sua operação semi-automatizada decidiu-se pela compensação do *baseline* através da medida de seu valor médio[20]. A área corrigida do sinal pode então ser obtida:

$$A_{corrigida}\,[Vs] = Area_{medida}\,[Vs] - (Baseline_{medido}\,[V] \times k_t\,[s]) \qquad (12)$$

Onde k_t é o tamanho da janela de aquisição, $40\ ns$ neste caso. A partir da área corrigida do pulso utilizou-se a Equação 9 para obter sua carga.

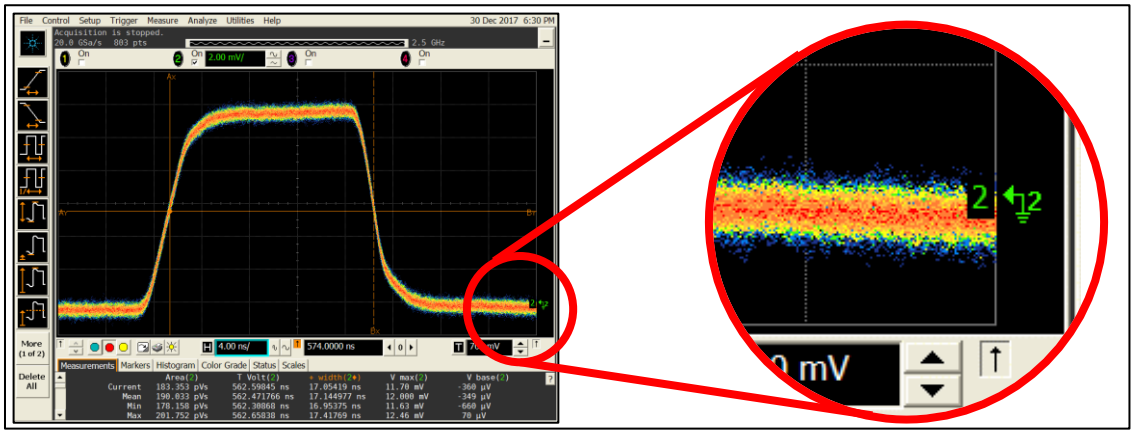


Figura 107 – Detalhe do *baseline* deslocado do sinal (Canal 0xF6; $wDAC$ 0x08; $aDAC$ 0x00).

[20] O manual de operação do osciloscópio descreve detalhadamente a metodologia utilizada para o cálculo do *baseline* do sinal. Resumidamente, trata-se do valor de pico do histograma composto por todos os pontos do sinal com tensão inferior a 10% de seu pico.

4.2.4. Avaliação das Saídas do Módulo Injetor

As seguintes grandezas, abaixo listadas, foram avaliadas a partir do conjunto de dados brutos para cada pulso produzido, e para todos os canais testados, de duas formas distintas: 1) como uma função da amplitude programada $aDAC$ e 2) como uma função da largura programada $wDAC$:

- Área corrigida [Vs] – Tensão integrada e corrigida pelo *baseline* (Eq. 12);
- Carga calculada [C] – Calculada a partir da área corrigida (Equação 9);
- Largura [s] – Largura do pulso a 50% de sua amplitude (FWHM);
- Temporização [s] – Tempo decorrido desde um referencial fixo (Trigger);
- Pico [V] – Tensão máxima registrada no pulso.

A análise dos resultados pelo escaneamento da amplitude considerou a variação das grandezas medidas como uma função da configuração de amplitude do pulso ($aDAC$) para vários valores fixos da configuração de largura do pulso ($wDAC$). Já a análise dos resultados pelo escaneamento da largura levou em conta a variação das grandezas medidas como sendo uma função da largura programada ($wDAC$), para diferentes valores de amplitudes ($aDAC$) fixos. Será visto que algumas grandezas são melhor interpretadas por alguma das abordagens.

4.2.4.1. Análise no Domínio da Amplitude

Foram produzidos, nesta análise, cinco gráficos – um por grandeza – para cada um dos 64 canais testados, totalizando 320 gráficos. Para efeito de exemplo encontram-se a seguir os gráficos da Figura 108, Figura 109, Figura 110, Figura 111 e Figura 112, todos relativos ao canal 0x05 e referentes a cada uma das grandezas analisadas: Área Corrigida, Carga Calculada, Largura, Atraso de Disparo e Tensão de Pico.

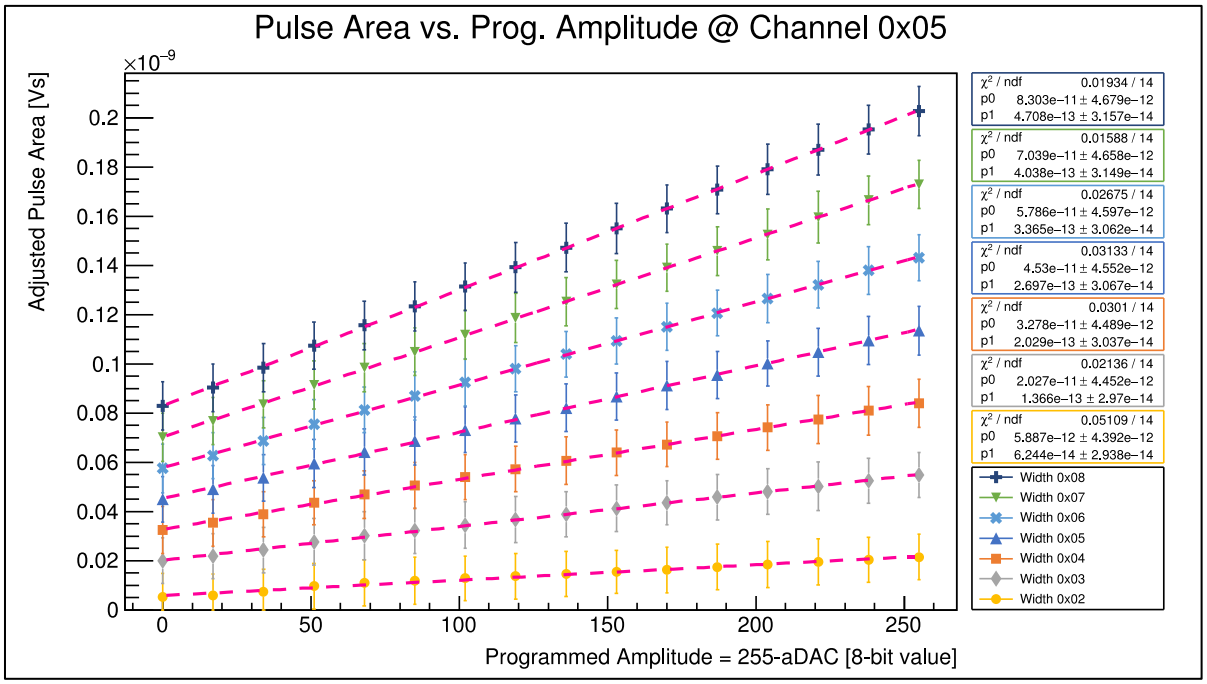


Figura 108 – Área corrigida do pulso em função de $aDAC$ no canal 0x05.

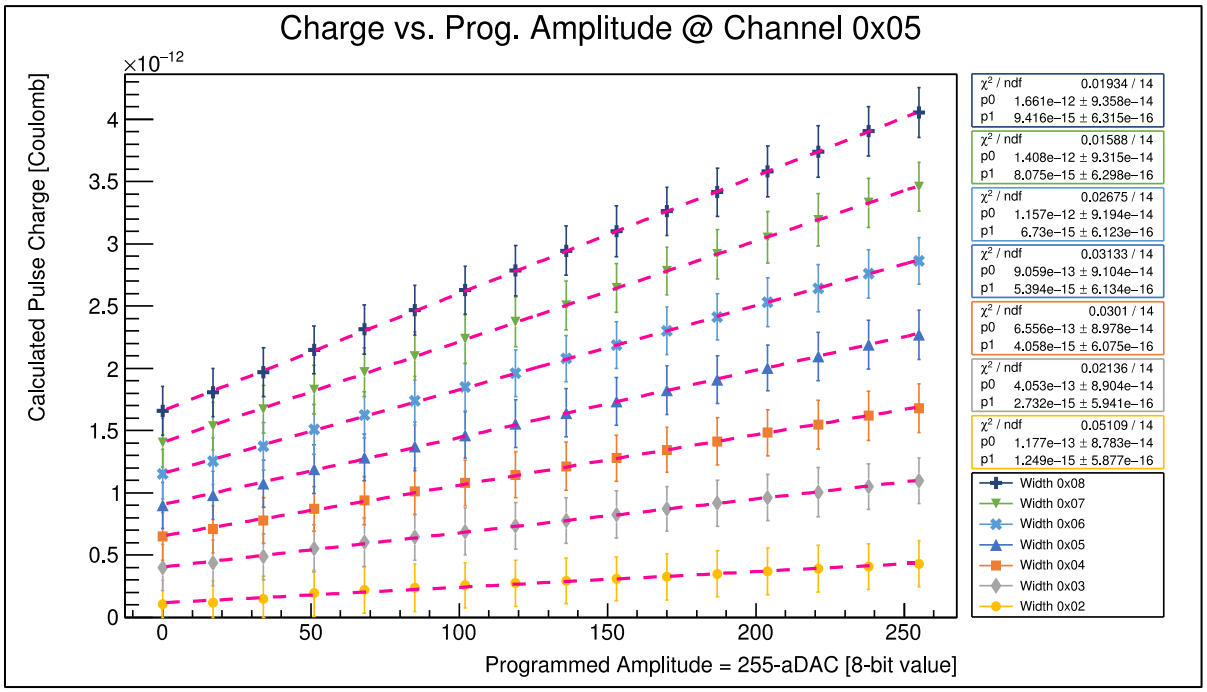


Figura 109 – Carga calculada do pulso em função de $aDAC$ no canal 0x05.

Nota-se grande semelhança entre os gráficos da Figura 108 e da Figura 109. Isto se deve à transformação linear (vide Equação 9) que relaciona a medida da área do pulso com sua quantidade calculada de carga. Resultam desta análise ajustes precisos ($X^2/ndf \ll 1$) o que a torna, portanto, válida e útil para estas grandezas.

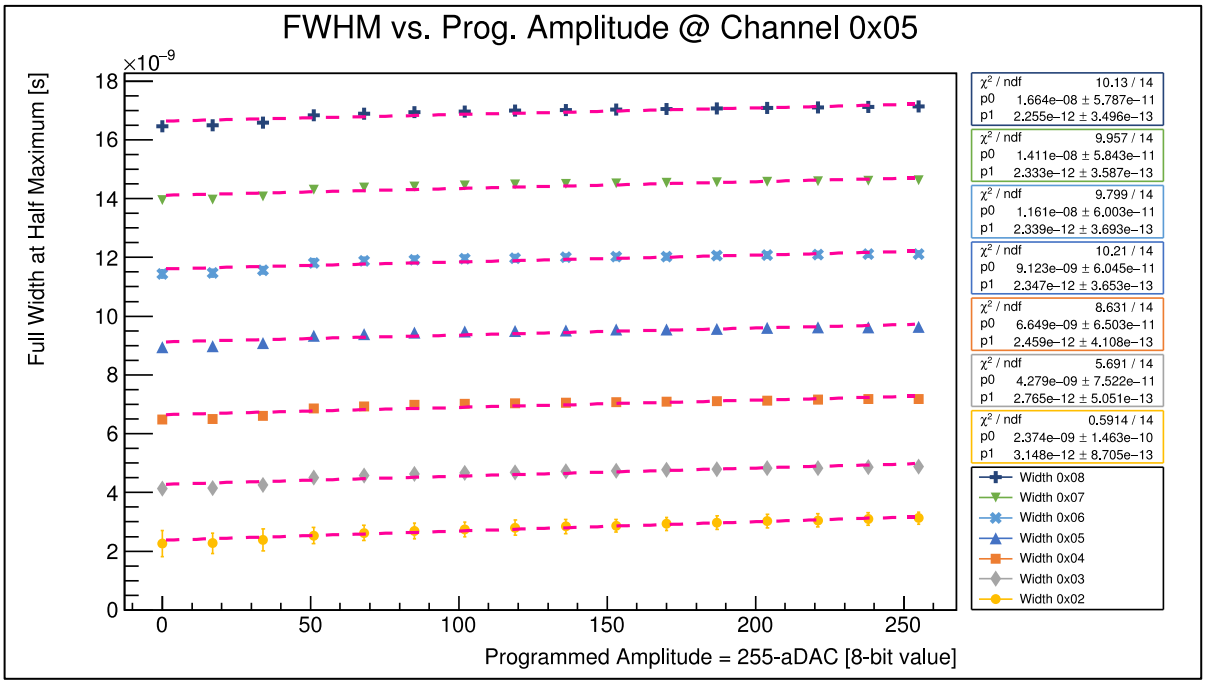


Figura 110 – Largura do pulso em função de $aDAC$ no canal 0x05.

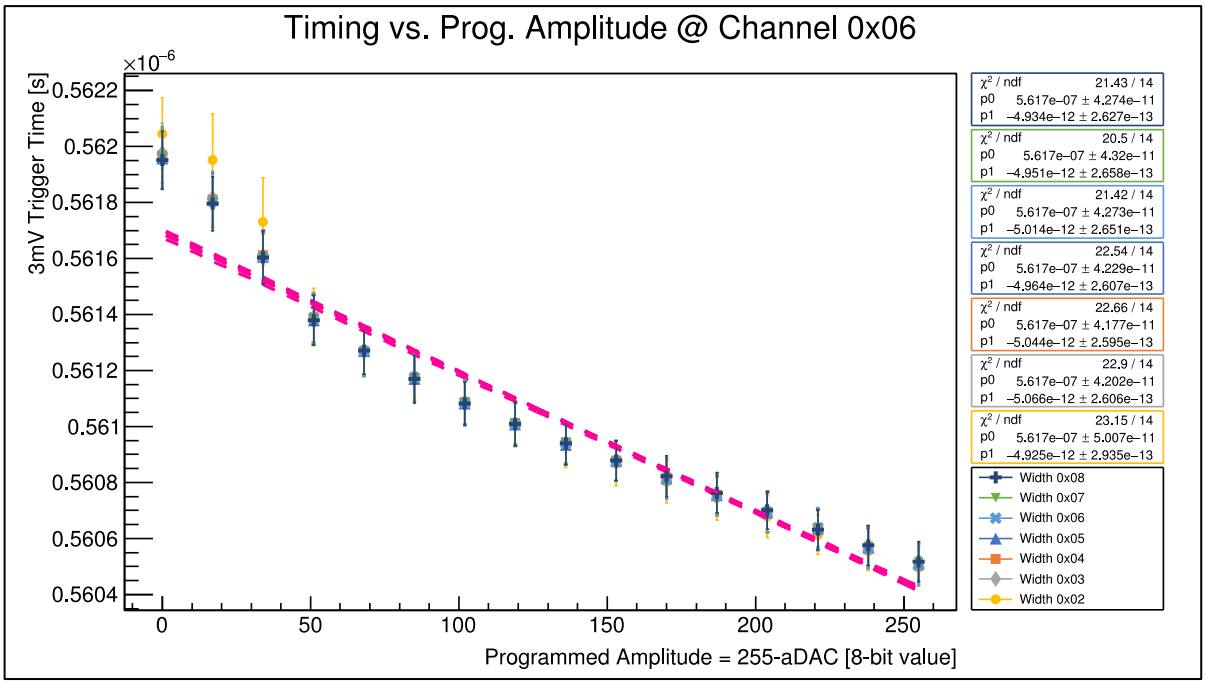


Figura 111 – Atraso de disparo do pulso em função de $aDAC$ no canal 0x06.

No gráfico da Figura 110 é possível notar uma pequena deriva na medida de largura do pulso à medida que $aDAC$ é incrementado, como já discutido em 4.1.2. A análise da variação de largura do pulso em função de $aDAC$ mostra-se útil, com boa precisão ($X^2/ndf < 1$) do ajuste linear. Já a análise do atraso de disparo pela mesma técnica mostrou-se pouco útil, apresentando $X^2/ndf > 1$.

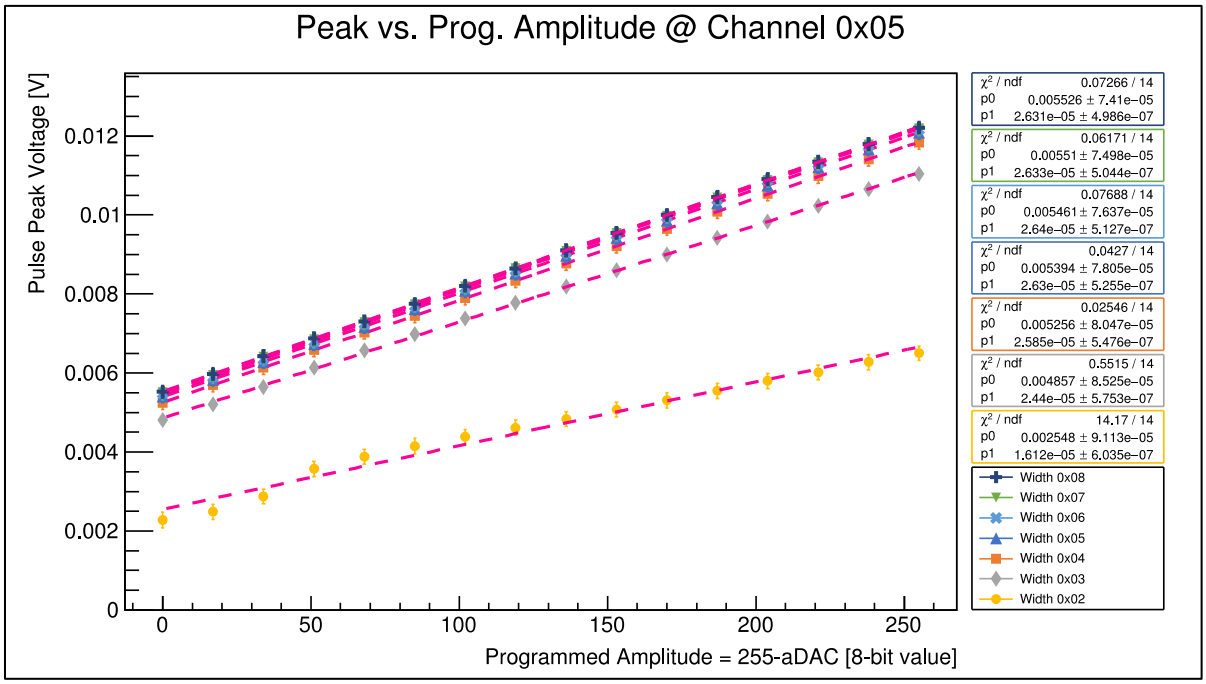


Figura 112 – Tensão de pico do pulso em função de $aDAC$ no canal 0x05.

A análise da variação de tensão de pico do pulso em função de $aDAC$ também se mostrou útil com ótima precisão ($X^2/ndf \ll 1$) nos ajustes lineares para todas as séries, exceto naquelas relativas a $wDAC$ igual $0x02$, devido à deformação do pulso que, nesta configuração, possui forma triangular e não chega a formar um *plateau*.

4.2.4.1.1. Extração dos Parâmetros de Ajuste

Como visto, cada gráfico produzido nesta análise está composto por sete séries de medidas, uma para cada configuração testada de largura de pulso. Foi então ajustado um polinômio de 1º grau (ajuste linear) a cada uma destas séries, extraindo-se dos $7 \times 5 \times 64 = 2.240$ polinômios ajustados seus três parâmetros geradores:

- p0: Coeficiente linear da reta ajustada;
- p1: Coeficiente angular da reta ajustada;
- Chi-Sq: Resultado do teste de qui-quadrado.

A partir do conjunto de parâmetros extraídos foi montado, para cada configuração testada de largura de pulso em cada grandeza, um conjunto de três histogramas. Cada um destes representa a distribuição dos parâmetros p0, p1 e Chi-Sq entre os 64 canais testados. Na Figura 113 encontra-se exemplificado o conjunto de histogramas para a distribuição dos parâmetros de ajuste da carga em função de $aDAC$ para $wDAC$ fixo e igual 0x02.

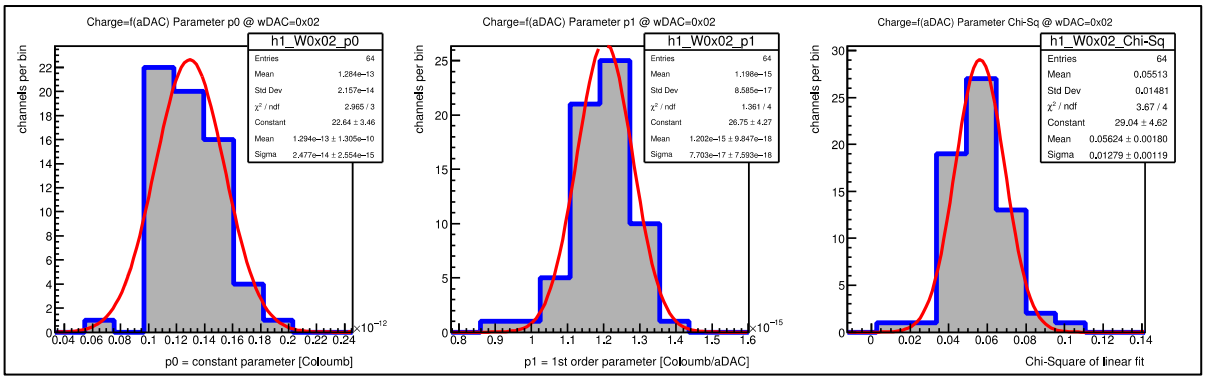


Figura 113 – Distribuição dos parâmetros de ajuste das funções de carga para $wDAC$ = 0x02.

Foram então produzidos sete conjuntos de histogramas por grandeza medida, um para cada $wDAC$ testado. Os 35 conjuntos de histogramas gerados nesta análise encontram-se para consulta no Apêndice K. Um ajuste gaussiano foi computado para cada histograma de modo a representar, conforme o caso, as distribuições de p0, p1 e Chi-Sq. Destes ajustes foram extraídos os três parâmetros geradores: Termo Constante, Média (μ) e Largura (σ).

A partir dos parâmetros μ e σ dos ajustes de p0 e p1 foi construída a Tabela 2, abaixo, onde são apresentadas equações gerais para calibração de carga, área, largura, temporização e tensão de pico em função do valor de $aDAC$ para várias configurações testadas de $wDAC$. Foi também incluída nesta tabela uma coluna com o valor máximo de X^2/ndf dos ajustes lineares comentados em 4.2.4.1. Uma escala de cores foi aplicada para indicar ajustes ótimos em preto para $X^2/ndf \leq 1$, bons em púrpura para $1 < X^2/ndf \leq 2$ e ruins em vermelho para $X^2/ndf > 2$.

Tabela 2 – Parâmetros de pulso em função de $aDAC$ para diferentes configurações fixas de $wDAC$.

Charge = p0 + p1*[aDAC]

wDAC [binary]	p0 Mean ± Sigma [Coulomb]	p1 Mean ± Sigma [Coulomb / aDAC]	Max_Err X^2/ndf
0x02	1.294E-13 ± 2.477E-14	1.202E-15 ± 7.703E-17	6.973E-03
0x03	4.124E-13 ± 1.597E-14	2.722E-15 ± 5.585E-17	3.414E-03
0x04	6.645E-13 ± 1.600E-14	4.056E-15 ± 6.192E-17	3.347E-03
0x05	9.186E-13 ± 1.933E-14	5.403E-15 ± 7.152E-17	3.392E-03
0x06	1.170E-12 ± 1.669E-14	6.745E-15 ± 6.534E-17	3.021E-03
0x07	1.424E-12 ± 1.777E-14	8.084E-15 ± 7.267E-17	3.873E-03
0x08	1.678E-12 ± 1.806E-14	9.424E-15 ± 8.047E-17	2.518E-03

Area = p0 + p1*[aDAC]

wDAC [binary]	p0 Mean ± Sigma [Volt*second]	p1 Mean ± Sigma [V*s / aDAC]	Max_Err X^2/ndf
0x02	6.250E-12 ± 1.262E-12	6.012E-14 ± 3.852E-15	6.973E-03
0x03	2.062E-11 ± 7.986E-13	1.361E-13 ± 2.793E-15	3.414E-03
0x04	3.330E-11 ± 8.224E-13	2.028E-13 ± 3.096E-15	3.347E-03
0x05	4.593E-11 ± 9.667E-13	2.702E-13 ± 3.576E-15	3.392E-03
0x06	5.856E-11 ± 8.593E-13	3.372E-13 ± 3.267E-15	3.021E-03
0x07	7.124E-11 ± 9.127E-13	4.042E-13 ± 3.634E-15	3.873E-03
0x08	8.400E-11 ± 9.471E-13	4.712E-13 ± 4.024E-15	2.518E-03

Width = p0 + p1*[aDAC]

wDAC [binary]	p0 Mean ± Sigma [second]	p1 Mean ± Sigma [s / aDAC]	Max_Err X^2/ndf
0x02	2.293E-09 ± 1.401E-10	3.141E-12 ± 1.728E-13	2.383E-01
0x03	4.317E-09 ± 1.418E-10	2.398E-12 ± 1.991E-13	7.002E-01
0x04	6.723E-09 ± 1.387E-10	2.027E-12 ± 2.277E-13	1.147E+00
0x05	9.201E-09 ± 1.446E-10	1.978E-12 ± 2.229E-13	1.242E+00
0x06	1.170E-08 ± 1.411E-10	1.973E-12 ± 1.914E-13	1.194E+00
0x07	1.421E-08 ± 1.390E-10	1.921E-12 ± 2.668E-13	1.275E+00
0x08	1.673E-08 ± 1.335E-10	1.916E-12 ± 1.883E-13	1.347E+00

Timing = p0 + p1*[aDAC]

wDAC [binary]	p0 Mean ± Sigma [second]	p1 Mean ± Sigma [s / aDAC]	Max_Err X^2/ndf
0x02	5.619E-07 ± 4.093E-10	-4.796E-12 ± 3.094E-13	2.111E+00
0x03	5.620E-07 ± 4.733E-10	-5.191E-12 ± 2.754E-13	2.849E+00
0x04	5.620E-07 ± 4.806E-10	-5.171E-12 ± 2.902E-13	2.472E+00
0x05	5.620E-07 ± 4.649E-10	-5.122E-12 ± 2.509E-13	2.479E+00
0x06	5.620E-07 ± 4.818E-10	-5.093E-12 ± 2.800E-13	2.414E+00
0x07	5.620E-07 ± 5.240E-10	-5.077E-12 ± 2.731E-13	2.344E+00
0x08	5.620E-07 ± 4.939E-10	-5.057E-12 ± 2.658E-13	2.680E+00

Peak = p0 + p1*[aDAC]

wDAC [binary]	p0 Mean ± Sigma [Volt]	p1 Mean ± Sigma [V / aDAC]	Max_Err X^2/ndf
0x02	2.822E-03 ± 4.866E-04	1.578E-05 ± 6.307E-07	2.703E+00
0x03	4.928E-03 ± 3.240E-04	2.477E-05 ± 3.460E-07	8.820E-02
0x04	5.262E-03 ± 3.327E-04	2.613E-05 ± 2.486E-07	2.155E-02
0x05	5.427E-03 ± 3.760E-04	2.642E-05 ± 1.781E-07	2.646E-02
0x06	5.398E-03 ± 3.554E-04	2.649E-05 ± 2.046E-07	3.781E-02
0x07	5.441E-03 ± 3.831E-04	2.652E-05 ± 1.692E-07	6.240E-02
0x08	5.489E-03 ± 3.919E-04	2.648E-05 ± 1.979E-07	3.027E-02

4.2.4.2. Análise no Domínio da Largura

A partir do mesmo conjunto de medições foi processada uma análise sob o ponto de vista da variação de largura do pulso. Para esta análise mais uma vez foram produzidos 320 gráficos, representando as grandezas analisadas em cada canal como uma função de $wDAC$ para várias configurações de $aDAC$.

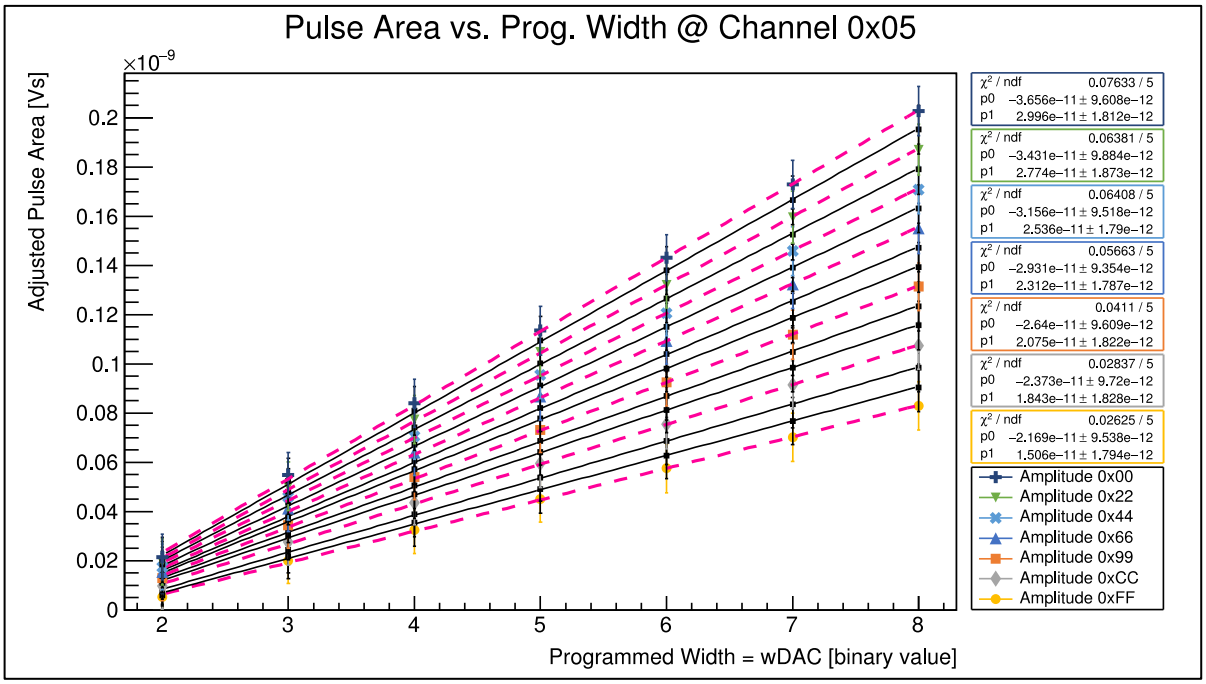


Figura 114 – Área corrigida do pulso em função de $wDAC$ no canal 0x05.

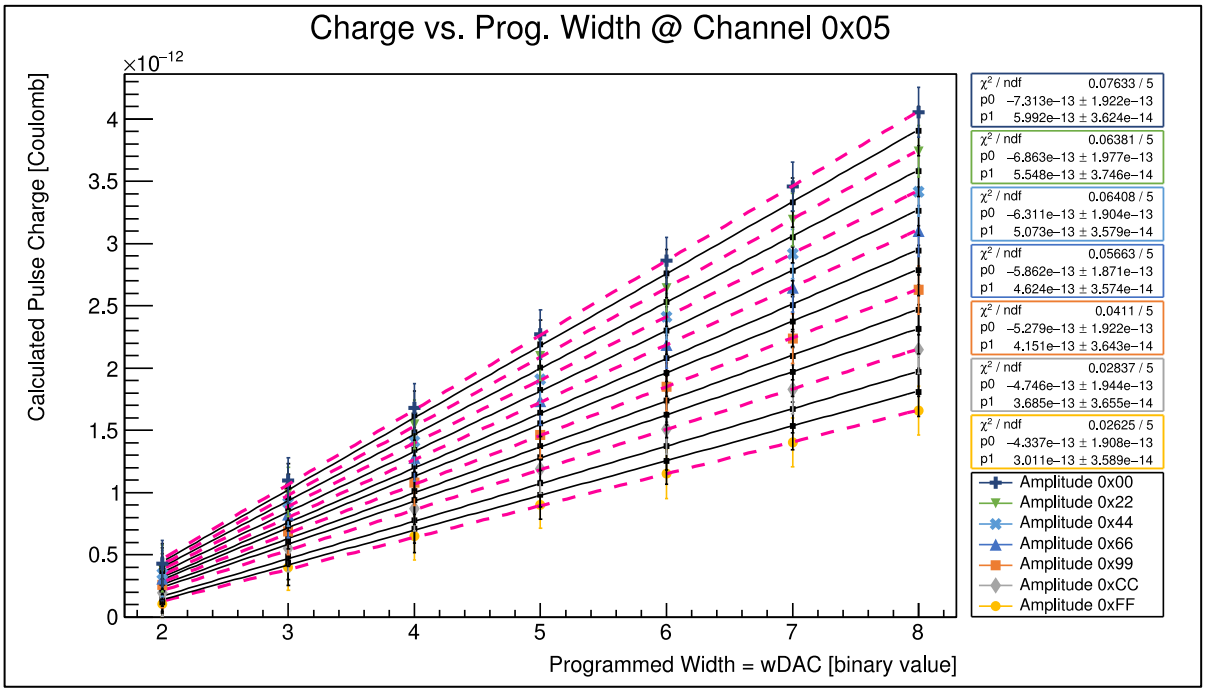


Figura 115 – Carga calculada do pulso em função de wDAC no canal 0x05

Mais uma vez é possível notar grande semelhança entre os gráficos de área do pulso (Figura 114) e de quantidade calculada de carga (Figura 115) devido à transformação linear (Equação 9) que relaciona estas medidas. Os ajustes lineares computados nesta análise mostraram-se precisos ($X^2/ndf \ll 1$), permitindo considerá-la válida e útil para a caracterização do Módulo Injetor.

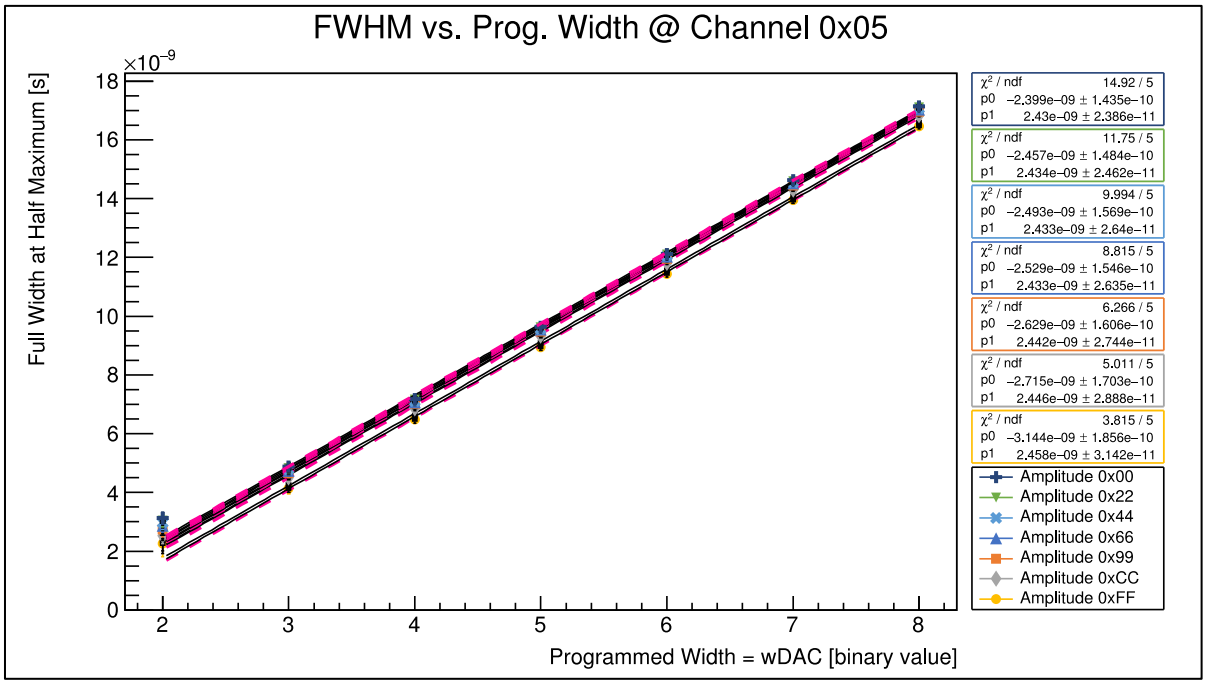


Figura 116 – Largura do pulso em função de $wDAC$ no canal 0x05.

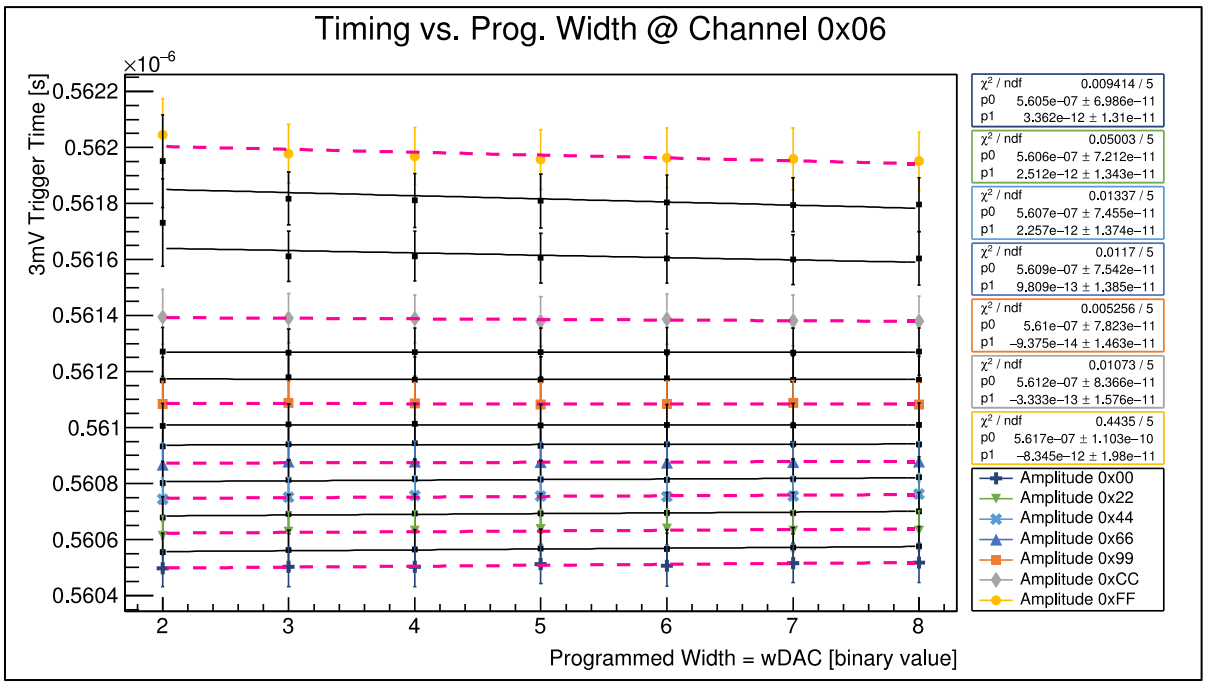


Figura 117 – Atraso de disparo do pulso em função de $wDAC$ no canal 0x06.

A análise da largura do pulso como uma função de $wDAC$ (Figura 116) mostrou-se ineficaz devido à baixa precisão ($X^2/ndf > 1$) dos ajustes lineares, mas permite observar qualitativamente o efeito de $wDAC$ na largura de pulso medida. Já a análise do atraso de disparo pela mesma técnica se mostrou útil, com $X^2 \,/\, ndf \ll 1$. No gráfico da Figura 117 é possível notar deriva no atraso medido em função de $aDAC$. Isto se deve a alterações da forma do pulso à medida que sua amplitude é alterada.

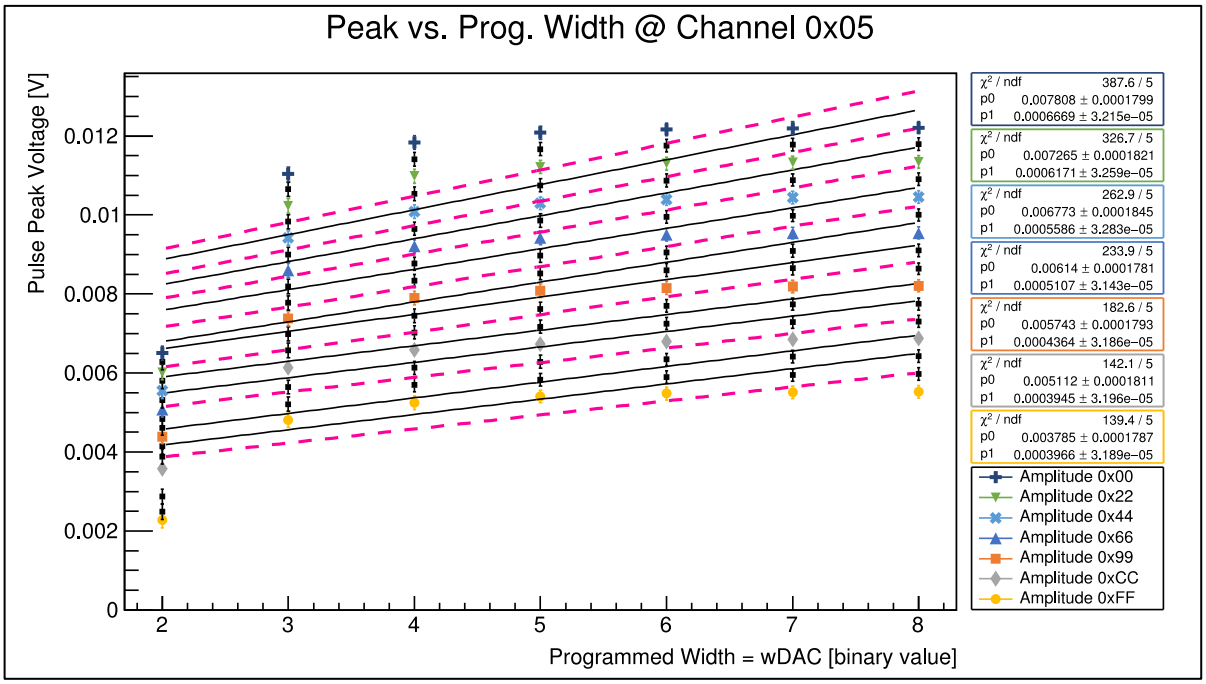


Figura 118 – Tensão de pico do pulso em função de $wDAC$ no canal 0x05.

A análise da tensão de pico do pulso em função de $wDAC$ não se mostrou útil devido à baixa precisão ($X^2 \,/\, ndf \ggg 1$) dos ajustes lineares. Isto se justifica pela falta de correlação entre a tensão de pico e a configuração $wDAC$.

4.2.4.2.1. Extração dos Parâmetros de Ajuste

Mais uma vez foram gerados conjuntos de histogramas com as distribuições dos parâmetros de ajuste linear e, a partir destes, a Tabela 3 para calibração em função de $wDAC$. A Figura 119 mostra um exemplo dos histogramas desta análise.

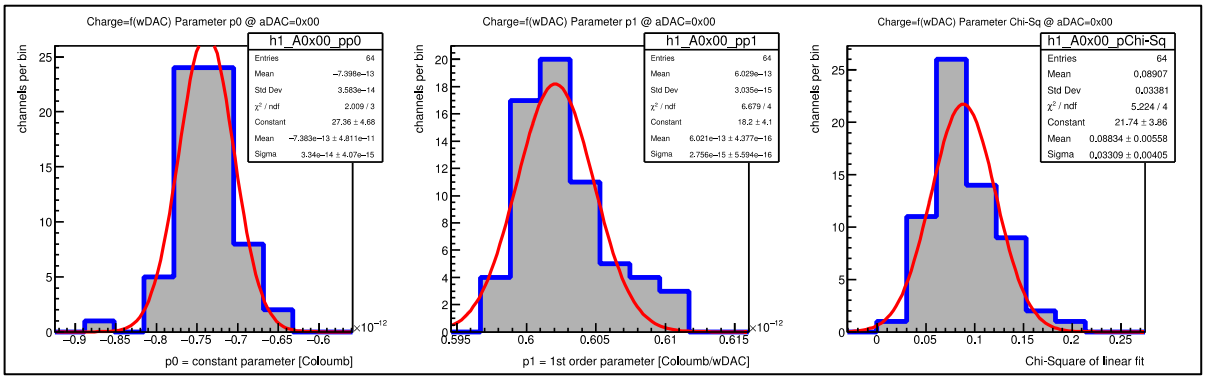


Figura 119 – Distribuição dos parâmetros de ajuste das funções de carga para $aDAC = 0x00$.

Tabela 3 – Carga do pulso em função de diferentes parâmetros de amplitude e largura.

Charge = p0 + p1*[wDAC]

aDAC	p0 Mean ± Sigma	p1 Mean ± Sigma	Max_Err
[binary]	[Coulomb]	[Coulomb / wDAC]	X^2/ndf
0xFF	-3.798E-13 ± 1.911E-14	2.571E-13 ± 2.490E-15	9.056E-03
0xEE	-4.115E-13 ± 2.521E-14	2.805E-13 ± 1.857E-15	9.945E-03
0xDD	-4.303E-13 ± 2.173E-14	3.027E-13 ± 2.413E-15	1.003E-02
0xCC	-4.285E-13 ± 2.400E-14	3.248E-13 ± 2.323E-15	7.727E-03
0xBB	-4.512E-13 ± 2.351E-14	3.475E-13 ± 2.311E-15	1.055E-02
0xAA	-4.760E-13 ± 2.502E-14	3.709E-13 ± 2.449E-15	1.340E-02
0x99	-5.023E-13 ± 2.652E-14	3.941E-13 ± 2.492E-15	1.471E-02
0x88	-5.287E-13 ± 2.735E-14	4.172E-13 ± 2.401E-15	1.500E-02
0x77	-5.582E-13 ± 2.838E-14	4.405E-13 ± 2.260E-15	1.875E-02
0x66	-5.846E-13 ± 3.080E-14	4.642E-13 ± 2.831E-15	1.997E-02
0x55	-6.120E-13 ± 3.050E-14	4.869E-13 ± 3.227E-15	2.256E-02
0x44	-6.368E-13 ± 3.109E-14	5.103E-13 ± 2.362E-15	2.325E-02
0x33	-6.628E-13 ± 3.041E-14	5.335E-13 ± 3.225E-15	2.576E-02
0x22	-6.907E-13 ± 3.614E-14	5.574E-13 ± 2.713E-15	2.825E-02
0x11	-7.162E-13 ± 3.565E-14	5.803E-13 ± 3.166E-15	3.172E-02
0x00	-7.383E-13 ± 3.340E-14	6.021E-13 ± 2.756E-15	3.749E-02

Area = p0 + p1*[wDAC]

aDAC	p0 Mean ± Sigma	p1 Mean ± Sigma	Max_Err
[binary]	[Volt*second]	[V*s / wDAC]	X^2/ndf
0xFF	-1.899E-11 ± 9.557E-13	1.285E-11 ± 1.245E-13	9.056E-03
0xEE	-2.057E-11 ± 1.261E-12	1.402E-11 ± 9.287E-14	9.945E-03
0xDD	-2.151E-11 ± 1.091E-12	1.514E-11 ± 1.207E-13	1.003E-02
0xCC	-2.161E-11 ± 1.259E-12	1.624E-11 ± 1.162E-13	7.727E-03
0xBB	-2.256E-11 ± 1.175E-12	1.738E-11 ± 1.156E-13	1.055E-02
0xAA	-2.380E-11 ± 1.251E-12	1.855E-11 ± 1.224E-13	1.340E-02
0x99	-2.512E-11 ± 1.326E-12	1.971E-11 ± 1.246E-13	1.471E-02
0x88	-2.643E-11 ± 1.367E-12	2.086E-11 ± 1.200E-13	1.500E-02
0x77	-2.791E-11 ± 1.419E-12	2.202E-11 ± 1.130E-13	1.875E-02
0x66	-2.940E-11 ± 1.607E-12	2.321E-11 ± 1.416E-13	1.997E-02
0x55	-3.060E-11 ± 1.525E-12	2.434E-11 ± 1.614E-13	2.256E-02
0x44	-3.184E-11 ± 1.554E-12	2.551E-11 ± 1.181E-13	2.325E-02
0x33	-3.314E-11 ± 1.520E-12	2.668E-11 ± 1.613E-13	2.576E-02
0x22	-3.454E-11 ± 1.807E-12	2.787E-11 ± 1.357E-13	2.825E-02
0x11	-3.581E-11 ± 1.783E-12	2.902E-11 ± 1.583E-13	3.172E-02
0x00	-3.688E-11 ± 1.676E-12	3.010E-11 ± 1.378E-13	3.749E-02

Width = p0 + p1*[wDAC]

aDAC	p0 Mean ± Sigma	p1 Mean ± Sigma	Max_Err
[binary]	[second]	[s / wDAC]	X^2/ndf
0xFF	-3.276E-09 ± 1.802E-10	2.470E-09 ± 4.229E-12	8.199E-01
0xEE	-3.238E-09 ± 1.855E-10	2.468E-09 ± 3.999E-12	9.343E-01
0xDD	-3.100E-09 ± 1.571E-10	2.465E-09 ± 4.204E-12	1.097E+00
0xCC	-2.849E-09 ± 1.181E-10	2.462E-09 ± 4.719E-12	1.359E+00
0xBB	-2.797E-09 ± 1.246E-10	2.460E-09 ± 3.757E-12	1.448E+00
0xAA	-2.748E-09 ± 1.094E-10	2.457E-09 ± 5.321E-12	1.619E+00
0x99	-2.707E-09 ± 1.155E-10	2.456E-09 ± 5.248E-12	1.836E+00
0x88	-2.671E-09 ± 1.159E-10	2.455E-09 ± 3.766E-12	2.161E+00
0x77	-2.645E-09 ± 1.099E-10	2.453E-09 ± 3.939E-12	2.359E+00
0x66	-2.615E-09 ± 1.109E-10	2.451E-09 ± 5.261E-12	2.576E+00
0x55	-2.606E-09 ± 1.045E-10	2.450E-09 ± 6.700E-12	2.686E+00
0x44	-2.577E-09 ± 1.006E-10	2.449E-09 ± 6.854E-12	3.149E+00
0x33	-2.562E-09 ± 9.106E-11	2.448E-09 ± 7.134E-12	3.258E+00
0x22	-2.532E-09 ± 9.957E-11	2.446E-09 ± 6.311E-12	3.898E+00
0x11	-2.517E-09 ± 1.047E-10	2.446E-09 ± 7.194E-12	4.000E+00
0x00	-2.495E-09 ± 1.220E-10	2.444E-09 ± 8.623E-12	4.519E+00

Timing = p0 + p1*[wDAC]

aDAC	p0 Mean ± Sigma	p1 Mean ± Sigma	Max_Err
[binary]	[second]	[s / wDAC]	X^2/ndf
0xFF	5.622E-07 ± 4.062E-10	-8.665E-12 ± 4.536E-12	2.202E-01
0xEE	5.622E-07 ± 5.108E-10	-9.776E-12 ± 5.256E-12	2.327E-01
0xDD	5.619E-07 ± 4.777E-10	-9.851E-12 ± 3.686E-12	2.158E-01
0xCC	5.616E-07 ± 4.017E-10	-5.109E-12 ± 3.796E-12	2.586E-01
0xBB	5.616E-07 ± 4.669E-10	-3.275E-12 ± 3.273E-12	1.625E-01
0xAA	5.613E-07 ± 3.984E-10	-1.962E-12 ± 2.865E-12	8.547E-02
0x99	5.612E-07 ± 3.607E-10	-9.837E-13 ± 2.624E-12	5.563E-02
0x88	5.612E-07 ± 3.609E-10	-4.155E-13 ± 2.002E-12	4.608E-02
0x77	5.611E-07 ± 4.175E-10	-2.926E-13 ± 2.422E-12	5.668E-02
0x66	5.610E-07 ± 4.171E-10	6.624E-13 ± 2.228E-12	4.507E-02
0x55	5.610E-07 ± 4.167E-10	8.228E-13 ± 2.270E-12	6.008E-02
0x44	5.609E-07 ± 4.620E-10	1.407E-12 ± 2.214E-12	5.982E-02
0x33	5.608E-07 ± 4.606E-10	1.613E-12 ± 2.062E-12	6.290E-02
0x22	5.608E-07 ± 4.606E-10	1.500E-12 ± 2.062E-12	5.864E-02
0x11	5.607E-07 ± 4.436E-10	1.965E-12 ± 2.284E-12	6.372E-02
0x00	5.607E-07 ± 4.429E-10	1.647E-12 ± 1.834E-12	8.050E-02

Peak = p0 + p1*[wDAC]

aDAC	p0 Mean ± Sigma	p1 Mean ± Sigma	Max_Err
[binary]	[Volt]	[V / wDAC]	X^2/ndf
0xFF	3.334E-03 ± 3.919E-04	3.355E-04 ± 4.260E-05	4.267E+01
0xEE	3.635E-03 ± 4.345E-04	3.592E-04 ± 4.737E-05	4.591E+01
0xDD	4.092E-03 ± 4.265E-04	3.732E-04 ± 3.997E-05	4.738E+01
0xCC	4.681E-03 ± 4.595E-04	3.492E-04 ± 4.116E-05	3.728E+01
0xBB	4.988E-03 ± 3.818E-04	3.639E-04 ± 3.729E-05	4.406E+01
0xAA	5.268E-03 ± 4.542E-04	3.855E-04 ± 4.051E-05	4.657E+01
0x99	5.589E-03 ± 4.229E-04	4.101E-04 ± 4.974E-05	5.109E+01
0x88	5.889E-03 ± 4.357E-04	4.359E-04 ± 4.742E-05	5.682E+01
0x77	6.192E-03 ± 4.323E-04	4.551E-04 ± 4.848E-05	6.405E+01
0x66	6.481E-03 ± 4.344E-04	4.855E-04 ± 5.962E-05	6.941E+01
0x55	6.757E-03 ± 4.270E-04	5.091E-04 ± 5.171E-05	7.369E+01
0x44	6.984E-03 ± 4.642E-04	5.358E-04 ± 5.645E-05	7.939E+01
0x33	7.305E-03 ± 4.273E-04	5.644E-04 ± 5.629E-05	9.220E+01
0x22	7.602E-03 ± 4.992E-04	5.827E-04 ± 5.506E-05	9.899E+01
0x11	7.902E-03 ± 5.452E-04	6.031E-04 ± 5.911E-05	1.091E+02
0x00	8.133E-03 ± 4.791E-04	6.306E-04 ± 6.600E-05	1.179E+02

4.2.4.3. Análise do Sincronismo Entre os Canais

Uma análise adicional foi processada a partir do conjunto de dados brutos da segunda sequência de testes com o objetivo de validar o plano de dimensionamento de trilhas descrito na Seção 3.7.1 de modo a quantificar as diferenças entre os atrasos de disparo do pulso (*timing*) dos canais testados. Como referencial de tempo foi utilizado o sinal de disparo (*trigger*) obtido de uma das portas SPARE do Módulo Injetor, como descrito em 4.2.1.

São apresentadas para várias configurações de $aDAC$ nos histogramas da Figura 120 as distribuições dos atrasos de disparo nos canais testados. Foram ajustadas a estes histogramas curvas Landau que permitem a análise quantitativa dos resultados.

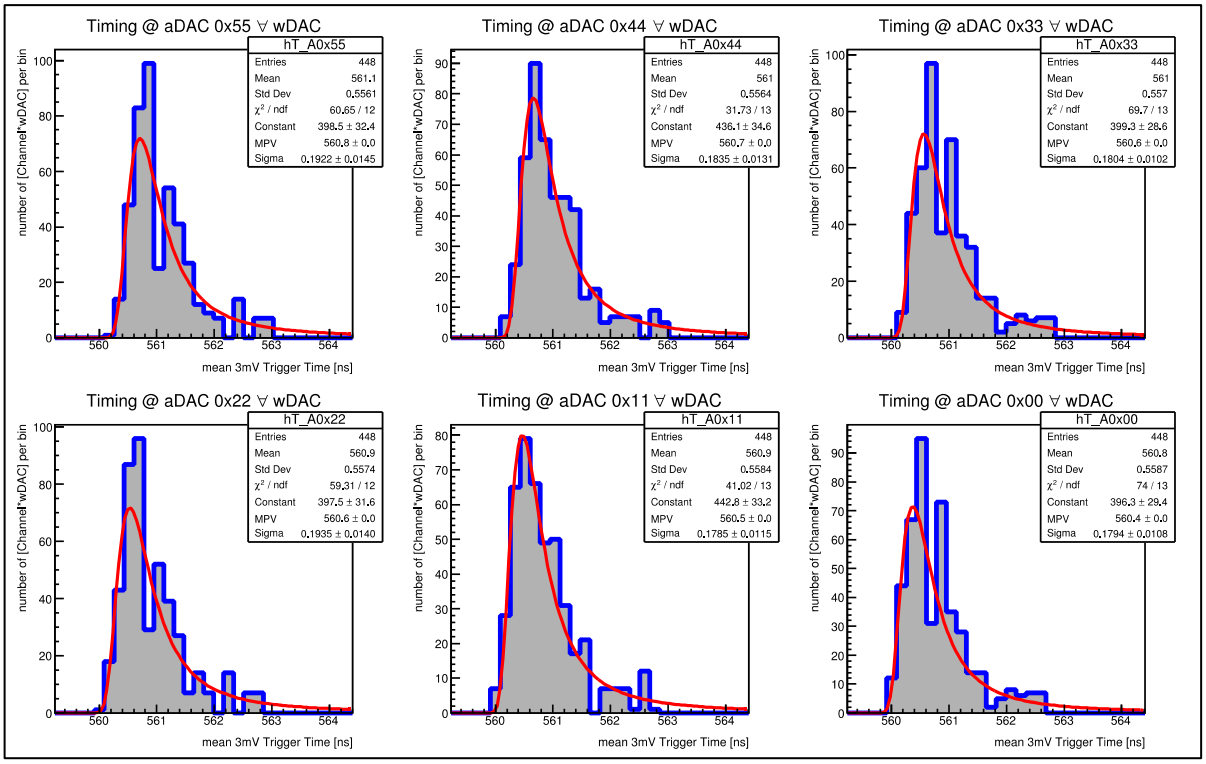


Figura 120 – Distribuições de atrasos intercanal para diversos valores de $aDAC$.

Na Figura 121 e na Figura 122 foram marcados sobre, respectivamente, os gráficos da Figura 111 e da Figura 117 os conjuntos de amostras que alimentam os vários histogramas da Figura 120.

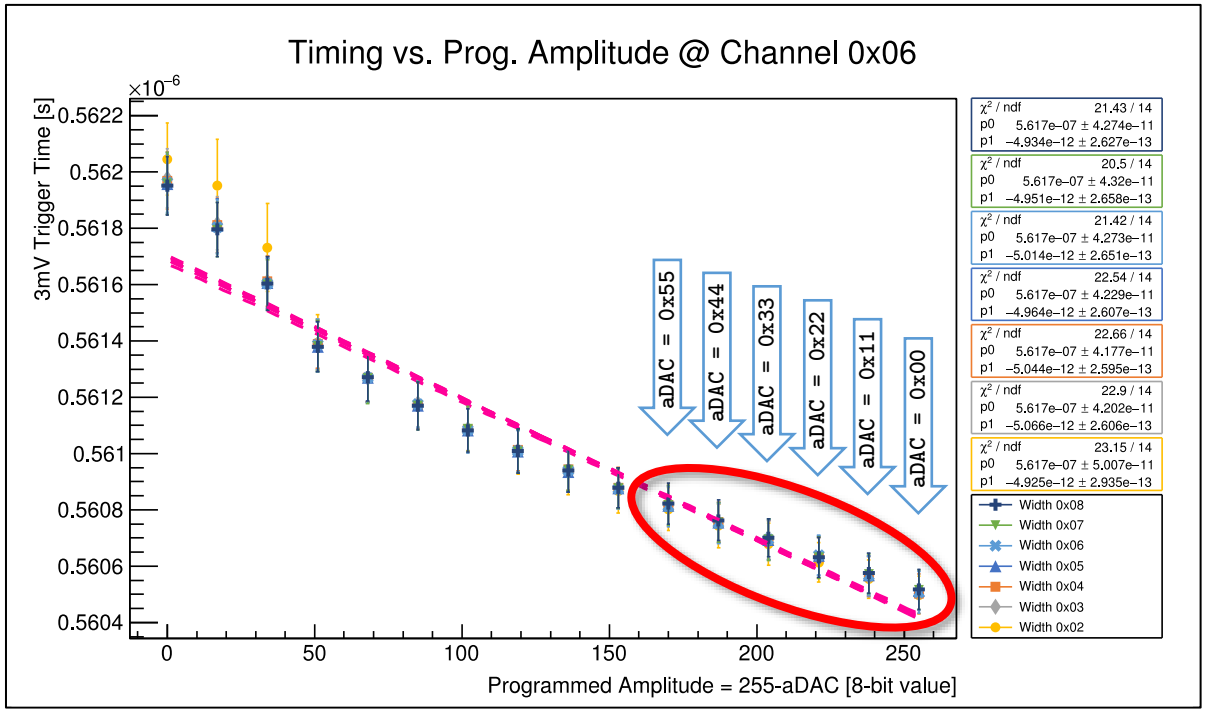


Figura 121 – Conjuntos de dados dos histogramas destacados..

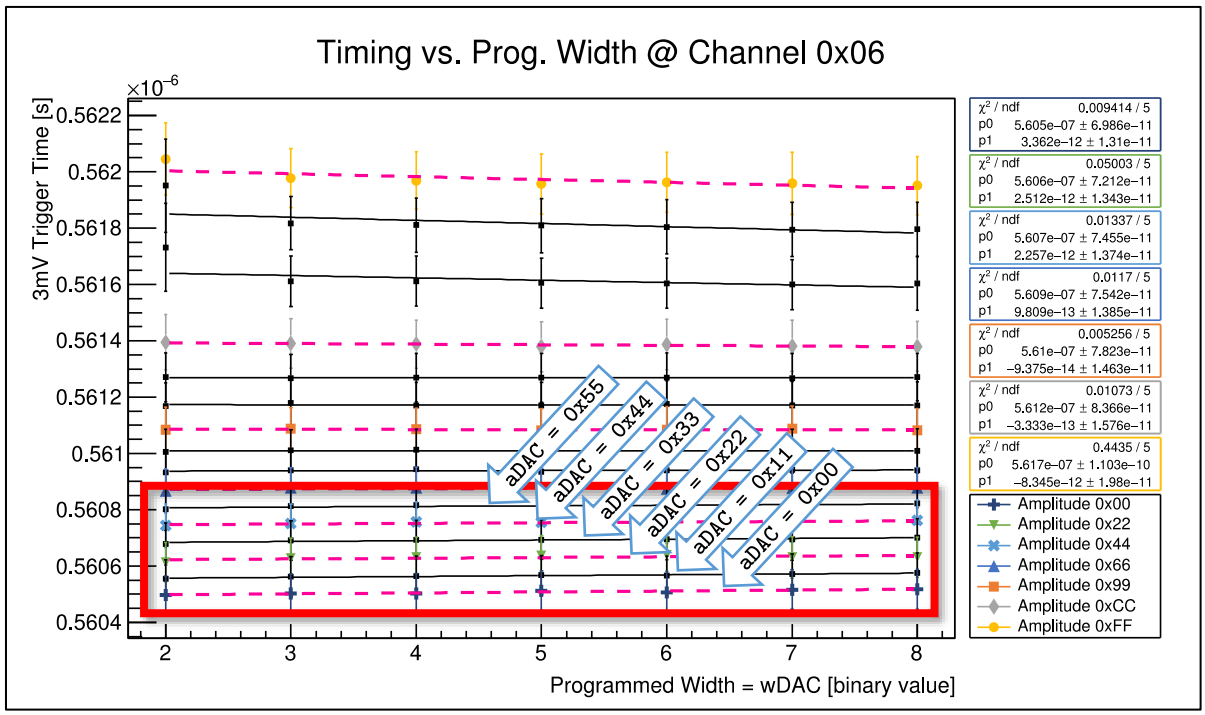


Figura 122 – Conjuntos de dados dos histogramas destacados.

Capítulo 5
Conclusões

A presente dissertação apresentou o desenvolvimento de um Dispositivo Injetor de Carga para o sistema de teste da eletrônica de *front-end* presente nas caixas de leitura, construídas para o *upgrade* do experimento *LHCb*.

Durante o período de pesquisa e desenvolvimento deste projeto optou-se por construir o dispositivo através de um Módulo de Controle e de oito Módulos Injetores, com 256 canais cada. O Módulo de Controle é, em essência, uma versão reduzida da *Master Board*, cujos principais componentes são o link óptico e os *Chips GBT* e *GBT-SCA*. O Módulo Injetor utiliza um FPGA como elemento central de processamento digital e 256 canais analógicos de processamento para a modulação da amplitude e o controle da forma do pulso. Os desafios tecnológicos enfrentados foram superados e os resultados de calibração obtidos são suficientes para concluir que o dispositivo desenvolvido alcançou com sucesso o objetivo inicialmente proposto.

5.1. Produção do Lote Piloto do Módulo Injetor

Um lote piloto com duas unidades do Módulo Injetor foi recebido e testado em maio de 2017 durante visita ao *NIKHEF*, em Amsterdam. Após implementadas as correções comentadas na Seção 3.8 as unidades deste lote apresentaram o comportamento funcional esperado, comprovando a validade conceitual do projeto. Uma das unidades foi enviada ao Brasil, encontrando-se atualmente no CBPF, e a outra permanece em Amsterdam para realização de testes complementares assim que a versão ***5*** do *ASIC* **PACIFIC** estiver disponível. Futuramente, as unidades podem servir de reserva para o Dispositivo Injetor de Carga.

5.2. Interpretação dos Resultados Obtidos

Foi possível concluir a partir da primeira sequência de testes que os pulsos produzidos pelo Módulo Injetor são capazes de excitar o *ASIC* **PACIFIC** e que a variação dos parâmetros *wDAC* e *aDAC* provoca variação do nível de tensão após o estágio de *Track & Hold* no *ASIC* em intervalo dinâmico suficiente para as condições de trabalho esperadas.

A segunda sequência de testes forneceu os conjuntos de equações de calibração apresentados na Tabela 2 e na Tabela 3 que permitem estimar diversas características do pulso produzido em função das configurações *aDAC* e *wDAC*. As equações representam, desta maneira, os dados de calibração do Módulo Injetor. A abordagem estatística utilizada permite generalizar estes dados de calibração para as demais 16 unidades que estão sendo produzidas.

Uma terceira sequência de testes com a versão ***5*** do *ASIC* **PACIFIC** ainda se faz necessária para fornecer resultados definitivos sobre a capacidade de o Módulo Injetor excitar o *ASIC* dentro do intervalo dinâmico esperado. Nesta sequência deverá ser analisada a saída digital do *ASIC* (e não a saída analógica *DEBUG*, como na primeira sequência) para diferentes configurações de pulsos testadas. Esta sequência de testes já foi programada, mas depende da disponibilização do *ASIC* **PACIFIC** versão ***5***, a qual vem sendo sofrendo atrasos sucessivos.

5.3. Continuidade do Sistema de Teste

Está prevista a construção de dois Dispositivos Injetores de Carga. Para isso, precisam ser produzidas mais dezesseis unidades do Módulo Injetor. A produção destas unidades está aguardando a conclusão dos testes complementares com a versão ***5*** do *ASIC* **PACIFIC**.

Com o desenvolvimento do Dispositivo Injetor de Carga, o *hardware* demandado pelo sistema de teste foi completado. Resta ainda o desenvolvimento de elementos de *software* que permitam operacionalizar este sistema. O painel de controle *SCADA* demonstrado em 3.9.2 pode ser expandido e acrescido do algoritmo de teste desejado. As ferramentas e métodos legados por este trabalho permitirão a realização desta tarefa.

5.4. Comissionamento do SciFi

Durante o comissionamento do novo detector de fibras cintilantes do *LHCb*, previsto para o 2º semestre de 2018, dar-se-á início à operação do teste propriamente dito, no qual a eletrônica de *front-end* das Caixas de Leitura recém-integradas deverá ser validada. Espera-se que o sistema de teste apresentado seja capaz de validar até quatro caixas de leitura por dia, completando este trabalho em aproximadamente quatro meses. O Dispositivo Injetor da Carga desempenhará, assim, papel crucial para o *upgrade* do experimento *LHCb*.

5.5. Produção Acadêmica Relacionada

No dia 28 de setembro de 2016 o Sistema de Teste foi apresentado na *TWEPP 2016 - Topical Workshop on Electronics for Particle Physics*, resultando na publicação do material reproduzido no Anexo A. ■

Referências Bibliográficas

[1] I. Aitchison and A. Hey, "Gauge theories in Particle Physics: A Practical Introduction (Book)", CRC Press, 2012.

[2] ATLAS and CMS Collaborations, "Combined Measurement of the Higgs Boson Mass in pp Collisions at s=7 and 8 TeV with the ATLAS and CMS Experiments", *Phys. Rev. Lett.,* p. 114, 2015. Disponível em: http://dx.doi.org/10.1103/PhysRevLett.114.191803 [Acessado: 18/02/2018].

[3] A. Sakharov, "Violation of CP invariance, C asymmetry, and baryon asymmetry of the universe", *Soviet Physics Uspekhi,* p. 34, 1991. Disponível em: https://doi.org/10.1070/PU1991v034n05ABEH002497 [Acessado: 18/02/2018].

[4] M. Klasen, M. Pohl e G. Sigl, "Indirect and direct search for dark matter", *Progress in Particle and Nuclear Physics,* p. 85, 2015. Disponível em: http://dx.doi.org/10.1016/j.ppnp.2015.07.001 [Acessado: 18/02/2018].

[5] The LHCb Collaboration, "The LHCb Detector at the LHC", *Journal of Instrumentation,* vol. 3, p. S08005, 2008. Disponível em: http://cds.cern.ch/record/1244506 [Acessado: 20/02/2018].

[6] INSPIRE, "INSPIRE, the High Energy Physics information system", [Online]. Disponível em: https://goo.gl/Z2eusN [Acessado: 18/02/2018].

[7] The LHCb Collaboration, "Evidence for CP violation in B±→K±π+π− and B±→K±K+K− decays", em *36th International Conference on High Energy*

Physics, Melbourne, Australia, 2012. Disponível em: http://cdsweb.cern.ch/record/1455471 [Acessado: 18/02/2018].

[8] The LHCb Collaboration, "Observation of J/ψp Resonances Consistent with Pentaquark States in Λ0b→J/ψK−p Decays", *Physical Review Letters,* vol. 115, no. 7, p. 117, 2015. Disponível em: https://doi.org/10.1103/PhysRevLett.115.072001 [Acessado: 20/02/2018].

[9] A. Sarti, "B Physics at LHC", in *Hadron Collider Physics Symposium (HCP2008)*, Galena, Illinois, USA, 2008. Disponível em: https://arxiv.org/abs/0809.0450 [Acessado: 18/02/2018].

[10] J. Christiansen, "Requirements to the L0 front-end electronics", CERN, Geneva, 2001. Disponível em: https://cds.cern.ch/record/691647 [Acessado: 18/02/2018].

[11] The LHCb Collaboration, "LHCb Tracker Upgrade Technical Design Report", CERN, Geneva, 2014. Disponível em: https://cds.cern.ch/record/1647400 [Acessado: 18/02/2018].

[12] The LHCb Collaboration, "Letter of Intent for the LHCb Upgrade", 2011. Disponível em: https://cds.cern.ch/record/1333091 [Acessado: 18/02/2018].

[13] J. Albrecht, C. Fitzpatrick, V. Gligorov e G. Raven, "The upgrade of the LHCb trigger system", em *WIT2014 Workshop on Intelligent Trackers*, Philadelphia, 2014. Disponível em: http://cds.cern.ch/record/1956613/ [Acessado: 20/02/2018].

[14] C. Elsasser, "The LHCb Silicon Tracker", *Journal of Instrumentation,* vol. 9, January 2014. Disponível em: http://dx.doi.org/10.1088/1748-0221/9/01/C01009 [Acessado: 20/02/2018].

[15] M. Tobin, "The LHCb Silicon Tracker", *Nucl. Instrum. Meth.,* pp. 168-172, 21 December 2013. Disponível em: http://dx.doi.org/10.1016/j.nima.2013.07.019 [Acessado: 18/02/2018].

[16] D. Eijk et al., "Radiation hardness of the LHCb Outer Tracker", *Nuclear Instruments and Methods,* vol. 685, pp. 62-69, 1 September 2012. Disponível em: https://doi.org/10.1016/j.nima.2012.05.033 [Acessado: 18/02/2018].

[17] The LHCb Outer Tracker group, "Performance of the LHCb Outer Tracker", *Journal of Instrumentation,* 2014. Disponível em: https://doi.org/10.1088/1748-0221/9/01/P01002 [Acessado: 18/02/2018].

[18] N. March e M. Karacson, "Radiation studies for the LHCb tracker upgrade", CERN, Geneva, 2014. Disponível em: http://cds.cern.ch/record/1662546 [Acessado: 18/02/2018].

[19] B. Beischer et al., "A high-resolution scintillating fiber tracker with silicon photomultiplier array readout", *Nuclear Instruments and Methods in Physics Research,* vol. 622, nº 3, pp. 542-554, 2010. Disponível em: http://dx.doi.org/10.1016/j.nima.2010.07.059 [Acessado: 18/02/2018].

[20] C. Joram, G. Haefeli e B. Leverington, "Scintillating Fibre Tracking at High Luminosity", em *SECOND INTERNATIONAL SUMMER SCHOOL ON INTELLIGENT SIGNAL PROCESSING FOR FRONTIER RESEARCH AND INDUSTRY*, Paris, 2014. Disponível em: http://dx.doi.org/10.1088/1748-0221/10/08/C08005 [Acessado: 18/02/2018].

[21] A. Bay et al., "Viability Assessment of a Scintillating Fibre Tracker for the LHCb Upgrade", CERN, Geneva, 2014. Disponível em: https://cds.cern.ch/record/1662525 [Acessado: 18/02/2018].

[22] I. Musienko, “Presentation: State of the art in SiPM's”, Geneva, 2011. Disponível em: https://indico.cern.ch/event/117424/contributions/1329259/ [Acessado: 06/2017].

[23] A. Vacheret et al., “Characterization and Simulation of the Response of Multi Pixel Photon Counters to Low Light Levels”, *Nucl. Instrum. Meth.,* vol. 656, nº 1, pp. 69-83, 11 November 2011. Disponível em: https://arxiv.org/abs/1101.1996 [Acessado: 18/02/2018].

[24] B. Dolgoshein et al., “Status report on silicon photomultiplier development and its applications”, *Nuclear Instruments and Methods,* vol. 563, nº 2, pp. 368-376, 15 July 2006. Disponível em: https://doi.org/10.1016/j.nima.2006.02.193 [Acessado: 18/02/2018].

[25] M. McClish et al., “Recent advances of planar silicon APD technology”, *Nuclear Instruments and Methods,* pp. 36-40, 1 November 2006. Disponível em: https://doi.org/10.1016/j.nima.2006.05.055 [Acessado: 18/02/2018].

[26] P. Buzhan et al., "Silicon photomultiplier and its possible applications", *Nuclear Instruments and Methods,* vol. 504, no. 1-3, pp. 48-52, 21 May 2003. Disponível em: https://doi.org/10.1016/S0168-9002(03)00749-6 [Acessado: 18/02/2018].

[27] P. Buzhan et al., "Timing by silicon photomultiplier: A possible application for TOF measurements", *Nuclear Instruments and Methods,* vol. 567, no. 1, pp. 353-355, 1 November 2006. Disponível em: https://doi.org/10.1016/j.nima.2006.05.142 [Acessado: 18/02/2018].

[28] D. Renker, "Geiger-mode avalanche photodiodes, history, properties and problems", *Nuclear Instruments and Methods,* vol. 567, no. 1, pp. 48-56, 1

November 2006. Disponível em: https://doi.org/10.1016/j.nima.2006.05.060 [Acessado: 18/02/2018].

[29] M. Danilov, "Novel photo-detectors and photo-detector systems", *Nuclear Instruments and Methods,* vol. 604, no. 1-2, pp. 183-189, 1 June 2009. Disponível em: https://doi.org/10.1016/j.nima.2009.01.208 [Acessado: 18/02/2018].

[30] K. Wyllie et al., "Electronics Architecture of the LHCb upgrade", CERN, Geneva, 2011. Disponível em: http://cds.cern.ch/record/1340939/files/LHCb-PUB-2011-011.pdf [Acessado: 18/02/2018].

[31] P. Moreira, J. Christiansen e K. Wyllie, "GBTX Manual", GBT PROJECT, Geneva, 2016. Disponível em: https://espace.cern.ch/GBT-Project/GBTX/Manuals/Forms/AllItems.aspx [Acessado: 18/02/2018].

[32] S. Bonacini et al., "GBT-SCA User Manual", Geneva, 2017. Disponível em: https://espace.cern.ch/GBT-Project/GBT-SCA/Manuals/Forms/AllItems.aspx [Acessado: 18/02/2018].

[33] A. Caratelli et al., "The GBT-SCA, a radiation tolerant ASIC for detector control and monitoring applications in HEP experiments", *Journal of Instrumentation,* vol. 10, nº 03, p. C03034, 2015. Disponível em: https://doi.org/10.1088/1748-0221/10/03/C03034 [Acessado: 18/02/2018].

[34] H. Verkooijen and W. Vink, "Master Board, technical manual", Nikhef, Amsterdam, 2015.

[35] PACIFIC Project Team, "PACIFIC: the readout ASIC for the SciFi Tracker of the upgraded LHCb detector", *Journal of Instrumentation,* vol. 11, February

2016. Disponível em: https://doi.org/10.1088/1748-0221/11/02/C02021 [Acessado: 18/02/2018].

[36] PACIFIC Project Team, "PACIFICr4b datasheet", 2017. Disponível em: https://edms.cern.ch/document/1761863/1 [Acessado: 18/02/2018].

[37] A. Sedra e S. Roberts, "Current Conveyor Theory", em *Analogue IC Design: The Current-Mode Approach*, 1990, pp. 93-124.

[38] A. Pellegrino, H. Verkooijen e W. Vink, "Read Out Box(ROB) interfaces", Nikhef, Amsterdam, 2016. Disponível em: https://edms.cern.ch/document/1484875/1.4 [Acessado: 18/02/2018].

[39] J. Cachemiche et al., "Readout board specifications for the LHCb upgrade", CERN, Geneva, 2012. Disponível em: https://edms.cern.ch/document/1251709/1 [Acessado: 17/02/2018].

[40] MiniDAQ Project Team, "LHCb Upgrade MiniDAQ Handbook", CERN, 2014. Disponível em: https://cds.cern.ch/record/1648422 [Acessado: 18/02/2018].

[41] Saturn PCB Design, Inc., "Saturn PCB Design Toolkit Version 7.05", [Online]. Disponível em: http://www.saturnpcb.com/pcb_toolkit.htm [Acessado: 20/02/2018].

[42] CERN, "ROOT a Data analysis Framework", [Online]. Disponível em: https://root.cern.ch/ [Acessado: 20/02/2018].

- FIM -

Anexo A

Publicação TWEPP 2016

Jinst

Published by IOP Publishing for Sissa Medialab

Received: *November 16, 2016*
Revised: *January 26, 2017*
Accepted: *February 16, 2017*
Published: *March 13, 2017*

Topical Workshop on Electronics for Particle Physics,
26–30 September 2016,
Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

2017 JINST 12 C03053

LHCb Scintillating Fiber detector front end electronics design and quality assurance

W.E.W. Vink,[a,1] **A. Pellegrino,**[a,b] **G.C.M. Ietswaard,**[a] **J.C. Verkooijen,**[a] **U. Carneiro**[c] **and A. Massefferi**[c] **on behalf of LHCb SciFi collaboration**

[a] *Nikhef, National institute for subatomic physics, Amsterdam, The Netherlands*
[b] *University of Groningen, Groningen, The Netherlands*
[c] *CBPF Brazillian Center for Research in Physics, Rio de Janeiro, Brasil*

E-mail: wilco.vink@nikhef.nl

Abstract: The on-detector electronics of the LHCb Scintillating Fiber Detector consists of multiple PCBs assembled in a unit called Read Out Box, capable of reading out 2048 channels with an output rate of 70 Gbps. There are three types of boards: PACIFIC, Clusterization and Master Board. The Pacific Boards host PACIFIC ASICs, with pre-amplifier and comparator stages producing two bits of data per channel. A cluster-finding algorithm is then run in an FPGA on the Clusterization Board. The Master Board distributes fast and slow control, and power. We describe the design, production and test of prototype PCBs.

Keywords: Front-end electronics for detector readout; Manufacturing

[1] Corresponding author.

© 2017 IOP Publishing Ltd and Sissa Medialab srl

doi:10.1088/1748-0221/12/03/C03053

Contents

1 Introduction **1**

2 Scintillating fiber detector front end electronics **1**

3 Quality control of the PCB manufacturing **5**
3.1 Design for Manufacturability 5
3.2 Design for Test 6
3.3 DfX results 7

4 End-item FE Tester **7**

5 Conclusions **8**

2017 JINST 12 C03053

1 Introduction

The LHCb detector will be upgraded during the Long Shutdown 2 (LS2) of the LHC in order to cope with higher instantaneous luminosities. Moreover, in order to increase the overall trigger efficiency, a 40 MHz readout scheme will be adopted in the on-detector electronics. The current tracking detectors downstream of the LHCb dipole magnet will be replaced by a Scintillating Fibre (SciFi) Tracker [1].

The Front-End (FE) electronics interfaces to the Silicon Photo Multipliers (SiPMs) on one side and to the experiment data-acquisition and control system on the other. It will consist of more than 4600 boards with more than 524,000 readout channels. In all the steps of design, validation and mass production, special care has to be taken to ensure that all boards are thoroughly tested. This article focuses on the strategy chosen to achieve this result, emphasizing the role of "*early design involvement*" of PCB manufacturers and assembly companies. First a general description of the SciFi electronics is given in section 2. Then a detailed description of the procedures put in place already during the design stage to ensure its manufacturability and testability is presented in section 3. Finally in section 4 an end-item test device is described, capable of performing the final validation of an entire readout unit with a series of functional tests.

2 Scintillating fiber detector front end electronics

The FE electronics interfaces to the SiPMs on one side and to the experiment data-acquisition and control system on the other. All SiPM signals are amplified, shaped and digitized in the 64-channel PACIFIC [2] ASIC; four PACIFIC ASICs are located on each PACIFIC Board. In order to reduce the data volume, the digitized channel data is routed to a flash-based IGLOO2 FPGA running a cluster-finding algorithm. Each Clusterization Board hosts two such FPGAs, processing the data

of 256 PACIFIC channels. The cluster data is then further sent to GBTX serializers located on the Master boards; four Clusterization Boards are connected to one Master Board. A Read Out Box (ROB) contains two Master Boards, eight Clusterization Boards, and eight PACIFIC Boards, and can service an entire SciFi module (2048 SiPM channels). The inter-connections between the three types of boards are realized through FMC connectors. The data serialized by the GBTX ASICs on the Master Board are shipped over optical fibers. An Housekeeping FPGA on the Master Board drives a Light Injection System (LIS) injecting light at the end of each fiber mat to provide SiPM gain calibration. The Master Board hosts 11 FeastMP DC-DC converters [3], providing four different voltages to the various devices in the ROB. Each ROB consumes about 110 W, and is individually cooled by demineralized water lines at 19°C.

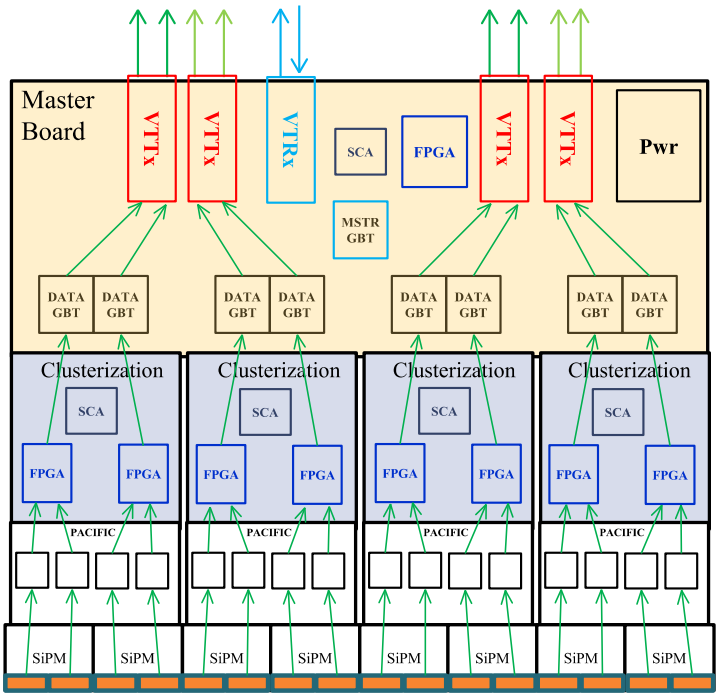


Figure 1. Block Diagram of half a ROB. The main components on each board are shown. The data flow is denoted by the green line.

The FE architecture is such that the cluster data of each SiPM are sent over a single fiber to the Back-End (BE) Electronics. Each ROB has also a complete interface for the distribution of bias voltages to the integrated circuits and to the SiPMs, of Timing and Fast Control (TFC) signals and of signals from and to the Experiment Control System (ECS). The communication with the TFC/ECS system goes via the GBT serial protocol: each Master Board hosts one Master GBT with an optical link connected to the LHCb SOL40 [6] detector control system. The entire clock tree is derived by deskewable clocks provided by the GBT. The clocks and fast-control commands are distributed to the Clusterization FPGAs and to the PACIFIC ASICs. For the slow control, five GBT-SCA ASICs [9] per half-ROB are used. The configuration of all devices is performed through

2017 JINST 12 C03053

SCA I^2C ports. The monitoring of the temperatures and of the SiPM bias voltages is based on the ADCs implemented in the SCAs. The SCA JTAG master is used to re-program all FPGAs.

All components are selected on the base of their radiation tolerance. The digitization of the SiPM signals is done by the 64-channel PACIFIC ASICs [2], the clustering algorithm is implemented in Microsemi M2GL090 FPGAs, and the data and control transmission uses the GBT chipset [4]. The power network implemented on the Master Board uses FeastMP DC/DC converters [3].

The LHCb SciFi detector consists of three tracking stations, each in turn consisting of four layers of scintillating-fibers modules. The active elements of a module are "mats" of scintillating material obtained stacking up six layers of round scintillating fibers with a diameter of 250 µm. Each module consists of 8 fiber mats. At one end of each fiber mat an 128-channel SiPM array is mounted [5]. Figure 2 shows a front view of the SciFi modules. The ROBs are attached at both ends of the modules. Each ROB consists of two boxes: a "Cold-Box" hosting 16 SiPM arrays (cooled down to −40°C) and a "FE-Box" containing the on-detector electronics previously described. The two are interconnected by flex cables. The SciFi on-detector electronics will consist of 256 ROBs, corresponding to 512 Master, 2048 Clusterization and 2048 PACIFIC Boards, for a total of about 524,000 readout channels.

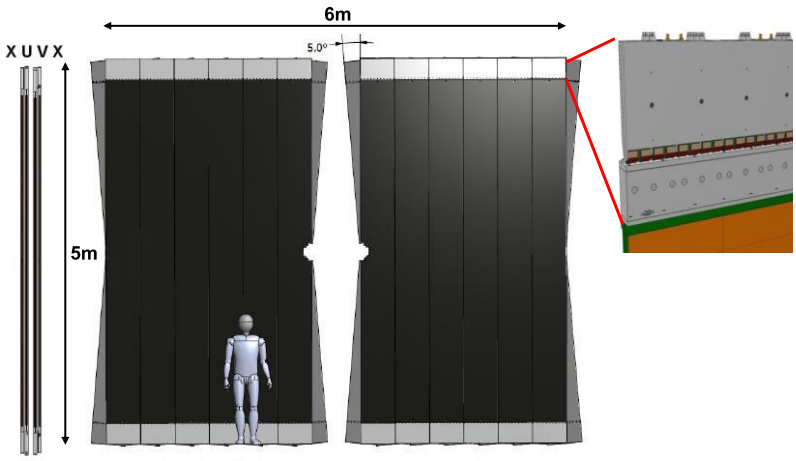


Figure 2. Overview of the SciFi fiber modules and Readout Boxes.

We produced and commissioned two ROB prototypes, including mechanical enclosure, cooling and shielding. We could successfully configure all devices on the ROB via the GBT serial links on the Master Board, using SOL40 firmware implemented on a prototype back-end electronics consisting of a MiniDAQ system based on the AMC40 architecture [7, 8]. In addition, we successfully demonstrated data transmission through the ROB, and could ship cluster data to the AMC40 MiniDAQ. We adopted the GBT "widebus" serial protocol at 4.48 Gbps, and a first implementation of the LHCb Tell40 DAQ framework in the AMC40, that processes incoming GBT data and produces 10GbE packets. We were able to perform a complete system test and demonstrated that the design of the SciFi FE architecture is ready to pass to the production phase.

An exploded-view drawing of a ROB is shown in figure 3, together with a picture of a half-ROB prototype. Clearly recognizable are the fiber-module end, the Cold-Box hosting the SiPMs, the flex

2017 JINST 12 C03053

cables connecting the SiPMs to the FE electronics, the various FE boards (Master, Clusterization and PACIFIC), and the metallic FE box providing cooling, grounding and shielding.

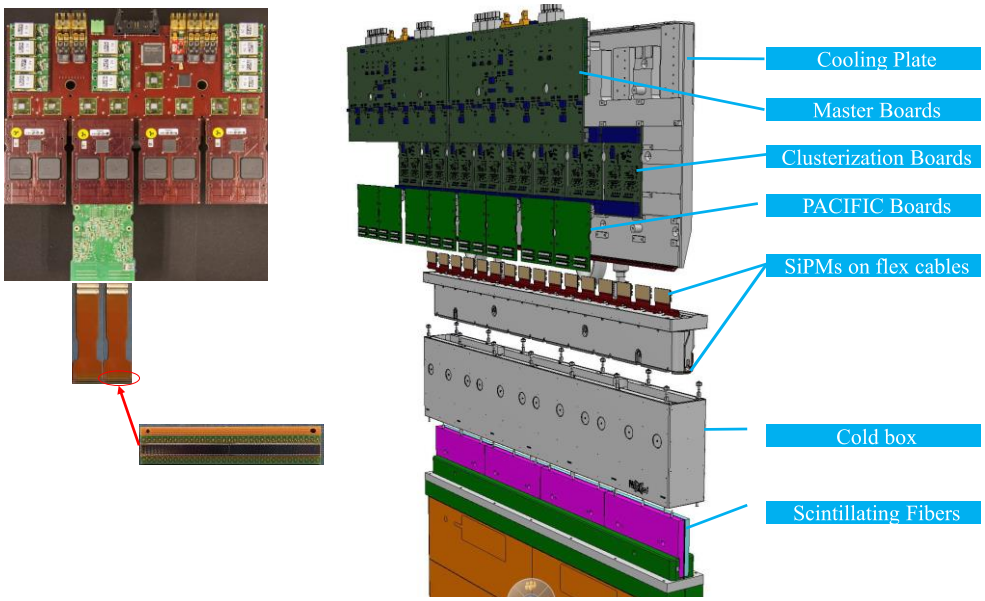


Figure 3. Image of the prototypes PCBs on the left and an exploded view of the Read Out Box including module, end piece, coldbox and electronics on a cooling frame.

As part of the prototype validation several cooling tests were performed. Figure 4 shows two infrared images of half a ROB; the temperature measured by three NTC sensors attached to some hot components are also shown. In the left panel one can see the temperature distribution without water flow in the right panel the temperature distribution with a water flow of 2 l/min.

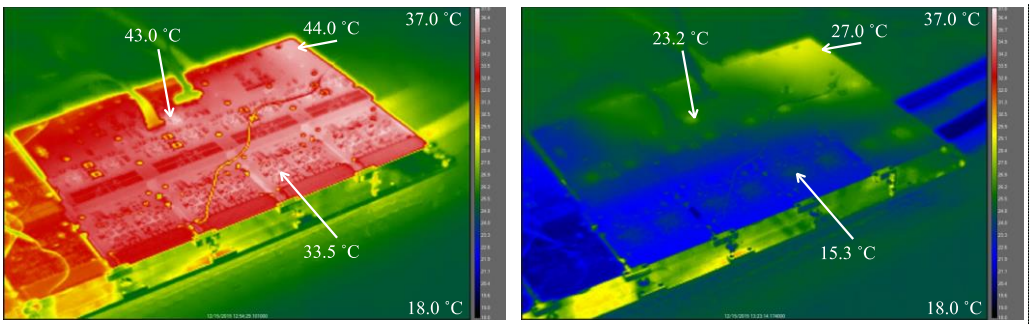


Figure 4. Thermal images taken during cooling test. Left panel: no water cooling. Right panel water flow of 2 l/min.

2017 JINST 12 C03053

3 Quality control of the PCB manufacturing

With 512 Master Boards, 2048 Clusterization Boards and 2048 PACIFIC Boards, to be produced and tested, a strategy to test all the boards, discard the faulty ones and ensure a satisfactory yield, need to be an integral part of the overall design process from the very beginning. The large number of components and nets on the PCBs make the design particulary complicated and error-prone. To give an example, nine 400 pins 0.8 mm pitch BGAs and a high number of *0402* capacitors and resistors are assembled on a single board. Moreover, all boards are expected to have a lifetime of ten to fifteen years in the LHCb detector.

While ECAD tools give the designer virtually unlimited design possibilities, the production processes at PCB manufacturer and assembly companies put several limitations of which the designer is not always aware, resulting in end items with a low production yield and limited lifetime. Therefore, the strategy chosen to meet these challenges emphasizes the role of "*early design involvement*" of PCB manufacturers and assembly companies. More specifically, for the design of the first prototypes of the Master and Clusterization boards we chose to implement two "Design For Excellence" (DfX) procedures [10]. A DfX typically consists of three phases during which design data is exchanged between the designer and the reviewer to guide the design during schematic entry, stack-up and layout, the first phase taking place before the design in the ECAD software starts, and the last phase taking place after the generation of production data. During each phase, the reviewer produces detailed reports describing the issues and giving design recommendations. The essential point is that the application of these DfX procedures guarantees that the production process will provide a given yield, quantified in terms of "slip through", i.e. the percentage of errors that could occur (mainly determined by those parts of the design that could not be made "testable").

Given the important role played in the SciFi electronics design process by the DfX procedures, and also considering that such procedures are rather uncommon in the design of electronics for particle and astro-particle physics experiments, they will be illustrated in more detail in the following sections. In particular, two DfX procedures will be considered separately: a "Design For Manufacturing" (DfM) one [11], to optimize the board design for the manufacturing and assembly process, and a "Design For Test" (DfT) one [12], to monitor the quality of the assembly process during and after the assembly of the boards. These two DfX procedures, DfM and DfT, are closely related and are applied in parallel during the design.

3.1 Design for Manufacturability

The main goal of the DfM process is to aid the design leading to a board that can be produced with the desired lifetime and yield. Figure 5 shows the three stages in which data is exchanged between the designer and the reviewer. Phase 0 is prior to the ECAD design: the reviewer delivers a set of design rules and the preferred data format for data exchange; the designer provides the reviewer with design specifications, block diagram(s) and the list of components. PCB materials (material properties, stack-up and trace width and clearances) and components suitable for lifetime and manufacturability are selected. After phase 0 the schematic entry starts and component footprints are created. Parts are placed on the preferred locations. The two outer copper layers are generated. In what is known as phase 1, schematics, outer copper layers with footprints and Bill Of Materials (BOM) are then

2017 JINST 12 C03053

reviewed. Footprints recommended by the component manufacturer are not always ideal for the assembly process. In some cases they need to be adapted to the assembly process. A trade off has to be established between components reliability and assembly process. The component placement determines the numbers of assembly stages; minimizing these can decrease the production cost. As an example: through-hole components are traditionally soldered by hand or by a (selective) wave soldering process; using Pin-In-Paste (PIP) components soldered during the reflow process supersedes the wave soldering process (larger pads become then necessary to hold the solder paste). The last phase can have several iterations, during which changes implemented by the designer as a result of the reviewer recommendations are checked. Once the complete layout is finished, a final check is done by the reviewer and a makeability report is before the manufacturing process starts.

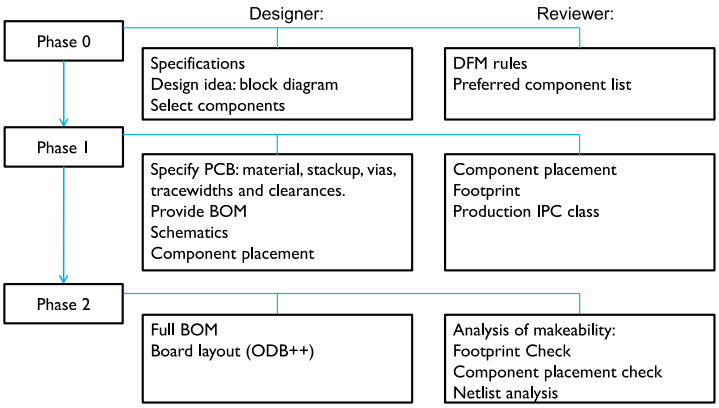


Figure 5. DfM process flow.

3.2 Design for Test

The DfT process aims at optimizing the design for testability. Its process flow is shown in figure 6. DfM and DfT run in parallel, and they share common data exchanged between the designer and the reviewer. The reviewer provides reports with the list of the tests that can provide the best test coverage at the lowest cost. In the last phase, a test strategy is chosen and the corresponding test coverage is reported. Several iterations of phase 2 are possible, to increase the coverage or decrease the cost. Typical tests that can be performed during or after production are:

- 3D Optical Automated inspection (3D-OAI). An optical PCB inspection will be done at three stages: the first one after applying the paste to check (against a pre-defined profile) whether the paste is correctly applied; the second one after the components are placed on the paste, eventually correcting erroneous placements of the parts; the third one after the reflow process to check the solder joints. 3D-OAI renders time-consuming and error-prone manual inspections superfluous.

- Automated X-ray Inspection (AXI). The solder joints of BGA parts are checked by X-ray inspection.

2017 JINST 12 C03053

- Flying Probe test (FPT). After the assembly, passive components, nets, and solder connections are tested with flying probes. To render this effective, as many testpoints as possible have to be foreseen to avoid using component pads, thus reducing the risk of damaging solder contacts.
- In Circuit Test (ICT). This is comparable to the FPT, but uses a custom-made needle card. It requires that all nets be accessible from the test side. Its main advantage with respect to the FPT is the shorter time needed.
- Extended Boundary Scan Test (EBST). All connections between boundary-scan capable devices are checked. Via passive and/or active loop-back board(s), connectors can be checked for defect solder contacts and broken nets.

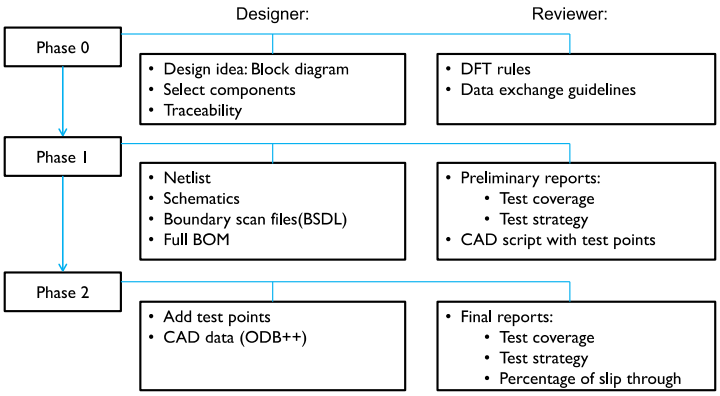


Figure 6. DfT process flow.

3.3 DfX results

Adopting DfM and DfT procedures in the SciFi prototype boards design resulted in boards that were free of production errors and could directly be used in functional tests without any need of electrical tests. The reviewer reports advised the test strategy shown in figure 7, with the expected coverage of each test (BGA X-ray inspection is a default test and therefore is not explicitly mentioned). The higher yield of the Clusterization board can be explained by its excellent boundary-scan coverage. The conclusion (small overall slip-through) is that if these procedures are followed for mass production, no dedicated electrical test of individual boards is required, and all boards can be directly assembled into a complete ROB, ready for the end-item test described in next section.

4 End-item FE Tester

End-item quality assurance of a complete FE Box requires a dedicated test setup capable of running a series of functional tests. An automatic FE Tester has been designed, consisting of a mechanical frame supporting several boards connected to the FE Box under test by flex cables. A schematic block diagram of the FE Tester is shown in figure 8. Control and data acquisition is performed with

2017 JINST 12 C03053

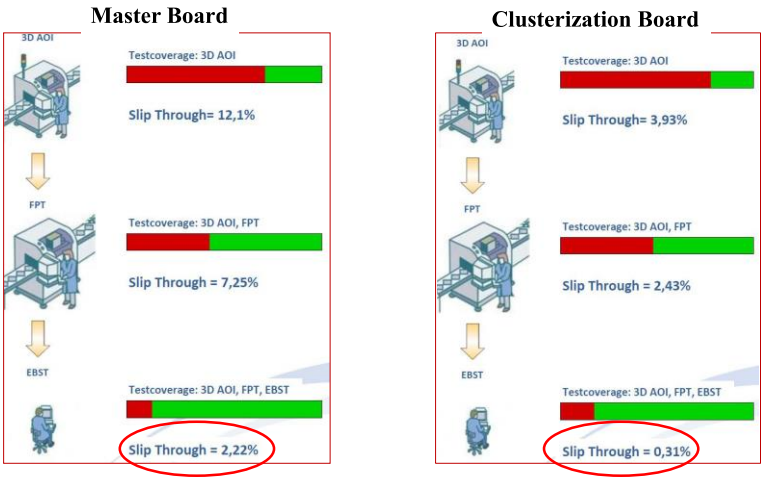


Figure 7. Advised test strategy and production slip through.

a MiniDAQ system, a generic system designed for the LHCb experiment. The FE Tester emulates the SiPM output pulses by a circuitry for each of the 2048 channels of a FE Box; due to the high density of components, this pulsed injector system consists of eight identical boards, each hosting a control FPGA. A control board hosting a GBT is the link between the MiniDAQ and the injector boards. A set of functional tests (channel-to-channel crosstalk, timing scans, threshold scans, etc.) will automatically be run controlled by procedures written in the standard LHCb WinCC SCADA software. All sixteen 4.8 Gb/s data optical links of the FE Box under test are read out by the MiniDAQ, and data are transferred to the host computer via 10 GbE and stored on disk, where data analysis jobs will produce a complete diagnosis, and store the results in a quality-assurance database. Beside aiding the quality assurance during mass production, the FE Tester will also be an indispensable diagnosis tool for the long-term mantainance and repair.

5 Conclusions

The LHCb detector will be upgraded during the Long Shutdown 2 (LS2) of the LHC in order to cope with higher instantaneous luminosities. The tracking detectors downstream of the LHCb dipole magnet will be replaced by the SciFi Tracker. The SciFi FE electronics will consist of more than 4600 boards with more than 524,000 readout channels. A complete strategy for the quality assurance of this electronics has been presented, consisting of several steps. In particular, the role of early-design involvement of the PCB manufacturing and assembly companies has been emphasized, and detailed Design-for-Manufacturing and Design-for-Test procedures, adopted already during the design stage, have been illustrated. The impact of these procedures on the first SciFi board prototypes have been presented, to show that boards free of production errors can be obtained. Quality control procedures have been identified that will allow mass production of Master boards and Clusterization Boards with yields of 97.78% and 99.69%, respectively. These high yields imply that the boards received from the producer(s) will not require additional tests before being

2017 JINST 12 C03053

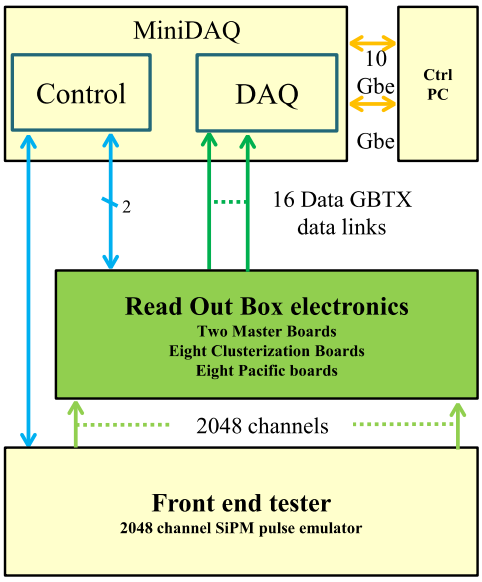


Figure 8. Block diagram of the SciFi full functional tester testing the ROB under test.

assembled into a complete FE Box. The quality assurance of each FE Box will be completed by an end-item test performed by a dedicated test setup, capable of running a series of functional tests that fully characterize the FE box, and of storing the results in a quality-assurance database for tracing.

References

[1] LHCb collaboration, *LHCb Tracker Upgrade Technical Design Report*, CERN-LHCC-2014-001.

[2] J. Mazorra et al., *PACIFIC: the readout ASIC for the SciFi Tracker of the upgraded LHCb detector*, 2016 *JINST* **11** C02021.

[3] S. Michelis et al., *FeastMP DCDC converter*, FeastMP Project Home Page.

[4] P. Moreira, *GBT Project, GBTX Manual*, GBT Project Home Page.

[5] X. Han, *LHCb upgrade: Development of a large scintillating fibre tracker*, in 10.1109/NSSMIC.2015.7581959.

[6] F. Alessio, *Timming and Fast Control aspects in a fully trigger-less PCIe-based readout architecture for the LHCb upgrade*, (2014) CERN-EDMS-1376144.

[7] F. Alessio and R. Jacobsson, *Readout Control Specifications for the Front-End and Back-End of the LHCb Upgrade*, (2014) LHCb-PUB-2012-017.

[8] K. Wyllie et al., *Electronics Architecture of the LHCb Upgrade*, (2013) LHCb-PUB-2011-011.

[9] A. Caratelli et al., *GBT-SCA, The slow control adapter for the GBT system*, GBT-SCA Project Home Page.

[10] J. Lehto et al., *Benefits of DfX in Requirements Engineering*, *Technology and Investment* **2** (2011) 27.

[11] A.J. Gadient et al., *Agility through information sharing: Results achieved in a production environment*, in proceedings of CE96, *Advances in Concurrent Engineering* (1996).

[12] G.R. Blackwell, *Basic Circuit Board Design: Overview and Guidelines*, *The Electronic Packaging Handbook* (1999).

2017 JINST 12 C03053

Apêndice A

Guia de Início Rápido do Módulo Injetor

Parte I – Operação em Standalone Mode

1) Faça as conexões de controle

Para operar o Módulo Injetor em modo autônomo você precisa conectar um PC através do adaptador I^2C para acessar seu banco de registradores internos. A interface de *debug* I^2C deve ser alcançada no conector SPARE através do pino Spare10, atuando como "SCL", Spare9, atuando como "SDA" e GND. Os pinos SDA e SCL devem ser conectados externamente a 3v3 através de resistores de pull-up de 10kΩ

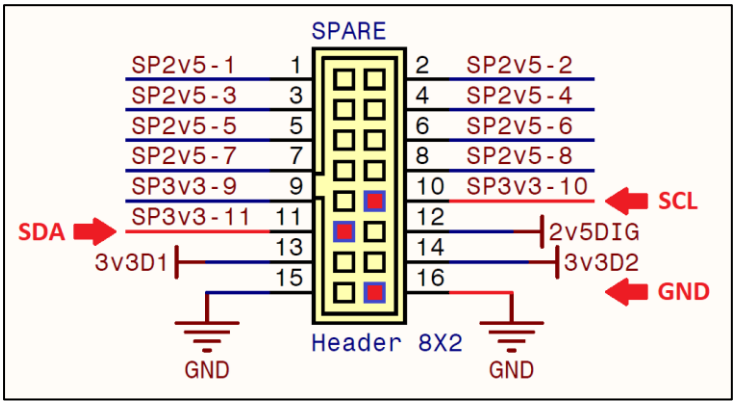


Figura 1 – Conector SPARE do Módulo Injetor

2) Selecione a fonte de *clock*

O cristal interno de 40MHz pode ser usado se você não quiser usar uma fonte de clock externa conectada ao par de pinos diferenciais HDMI 10 e 12. O interruptor "USERBIT0" para selecionar a fonte de relógio primária. USERBIT0 = '0' significa "Relógio interno ativo"; USERBIT0 = '1' significa "Relógio HDMI externo ativo". Cuidado: você seleciona a fonte do relógio principal. Em algumas circunstâncias, o circuito PLL pode alternar a fonte do relógio efetivamente em uso.

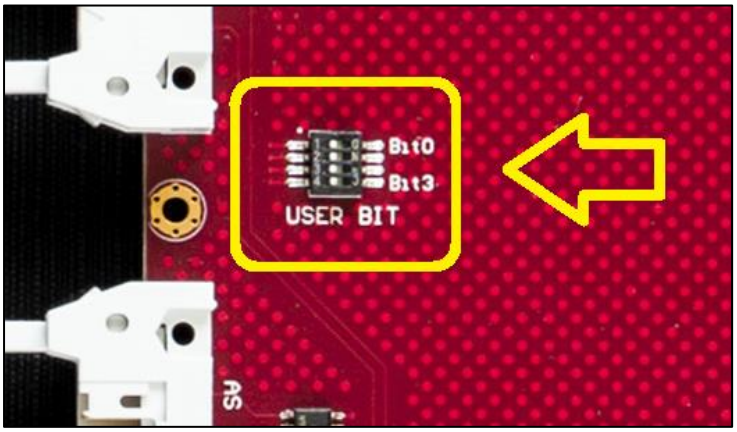


Figura 2 – Chaves USERBIT na superfície da PCB do Módulo Injetor

3) Selecione a interface de escrita da memória

Várias interfaces estão presentes para acesso ao banco de registro interno. Você pode usar todas elas simultaneamente para comandos de leitura, mas apenas uma pode ser utilizada por vez (e deve ser selecionada) para executar comandos de gravação. O interruptor "USERBIT2" seleciona a interface de gravação da memória. USERBIT2 = '0' significa" HDMI_I2C ativo"; USERBIT2 = '1' significa "DEBUG_I2C ativo". Para operar através do Módulo de Controle você deve selecionar "HDMI_I2C ativo".

4) Escreva em alguns registradores de controle

Após um ciclo de energia o Módulo Injetor inicializa com o valor 0x00 em quase todos os registradores internos, e isto significa "sem sinal em todas as saídas". Então, para ver um sinal você deve escrever em alguns registradores. Os comandos mínimos de gravação para despertar um canal encontram-se listados abaixo:

» Máscara On-Off: Mude para "1" o bit (ou bits) correspondente ao(s) canal(s). "1" significa ON; "0" significa OFF. Por exemplo, para ligar o canal 7, escreva 0x40 no registro 0x0800;

» Largura do pulso: Escreva a largura desejada no registrador STOP correspondente ao canal desejado. Por exemplo, para definir a largura do canal 7 para "0x04" escreva 0x04 no registrador 0x0200. Lembre-se que os registros START são iguais a zero após um ciclo de energia. Portanto, o valor de STOP é igual à largura neste momento;

» Taxa de repetição (Trigger Prescaler): Sem um comando externo de disparo, o Injetor pode gerar internamente este comando como um submúltiplo do relógio "principal" (por padrão, 40MHz). Você deve escolher a taxa de repetição escrevendo o registrador 0x0881. Por exemplo, para disparar a 1MHz, escreva 0x27 no registrador 0x0881;

» Ajuste a amplitude: Escreva o(s) valor(es) hexadecimais desejado(s) no(s) registrador(es) $aDAC$ correspondente(s). Observe que este é um valor invertido (0x00 significa máximo e 0xFF mínimo) e que o mapa de registradores de amplitude possui um esquema de endereçamento de correspondência indireta.

Para escrever nestes registradores (se você estiver usando o adaptador I^2C-*USB* fornecido) configure a porta COM para 9.600 bps; 8-bits; sem paridade; 1 bit de parada; controle de fluxo Xon/Xoff e inicie uma sessão no terminal serial. Após a rotina de configuração, você verá uma tela como mostrado na Figura 3. Para escrever digite "W" seguido do endereço de registro de 16 bits (4 símbolos hexadecimais) e um ENTER. Em seguida, digite o valor de 8 bits (2-hex) seguido de um ENTER.

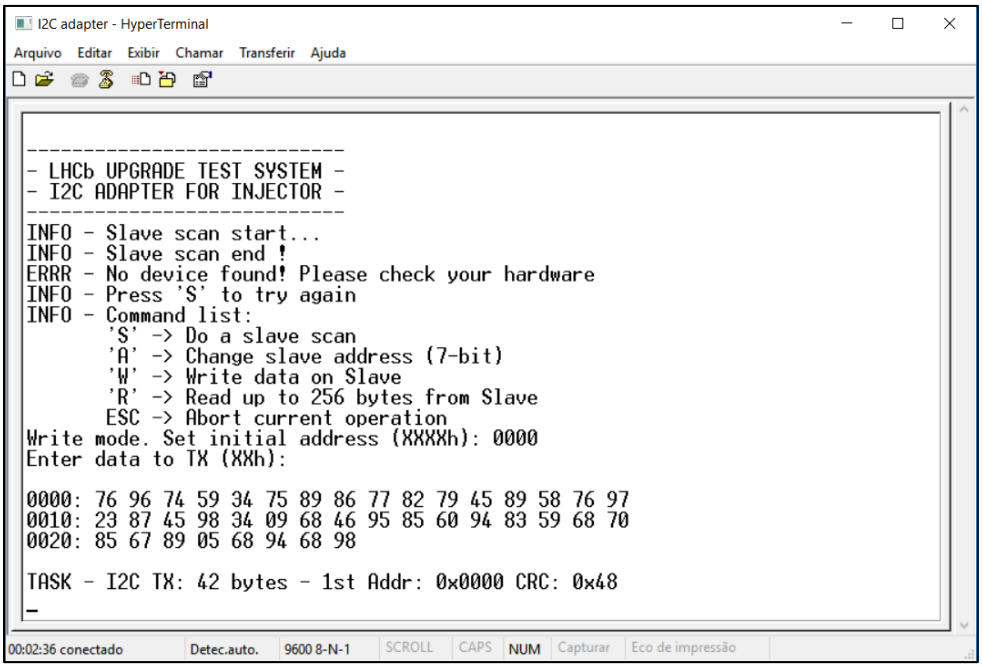


Figura 3 – Interface de usuário do adaptador I^2C-*USB* (no Windows HyperTerminal)

Parte II – Operação em MiniDAQ Mode

1) Conecte o módulo injetor ao módulo de controle através do cabo *HDMI*

O conector HDMI padrão presente no Módulo Injetor fornece acesso a todos os sinais necessários para seu controle. Através do cabo HDMI, o Módulo de Controle transmite quatro linhas diferenciais para controle rápido, um barramento I^2C bidirecional para controle lento, dois bits multiuso GPIO e uma linha de alimentação. Utilize um bom cabo HDMI, que possua todos os pinos conectados entre as extremidades.

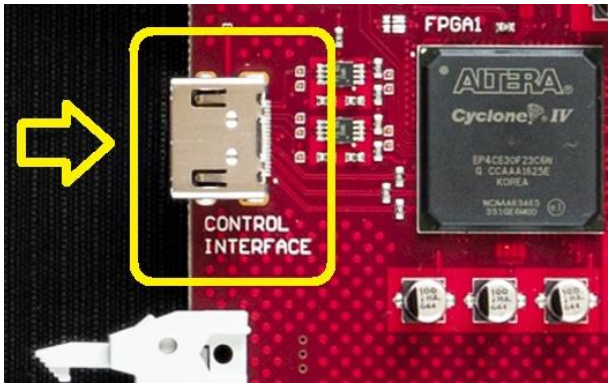


Figura 4 – Conector HDMI na superfície da PCB do Módulo Injetor

2) Selecione a fonte de *clock*

Para sincronizar todos os Módulos Injetores que estejam operando em conjunto uma fonte de relógio externo comum deve ser conectada ao par diferencial dos pinos 10 e 12 no HDMI. O Módulo de Controle fornece até oito saídas de relógio ajustáveis, uma em cada conector HDMI, todas derivadas do chip GBT e com alinhamento de fase individual. Então, tudo o que precisa ser feito é conectar todos os oito Módulos Injetores ao mesmo Módulo de Controle utilizando cabos HDMI de mesmo comprimento. Em seguida, use o interruptor "USERBIT0" para selecionar a fonte de relógio primária. USERBIT0 = '0' significa "Relógio interno ativo"; USERBIT0 = '1' significa "Relógio HDMI externo ativo". Cuidado: você seleciona a fonte do relógio principal. Em algumas circunstâncias, o circuito PLL pode alternar a fonte do relógio.

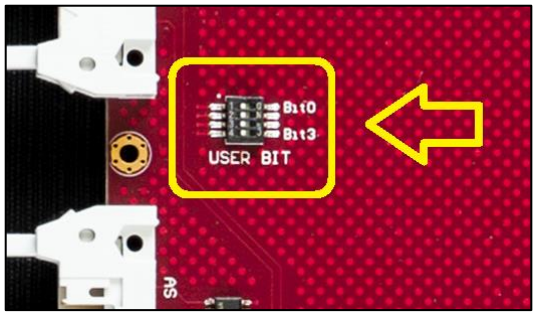


Figura 5 – Chaves USERBIT na superfície da PCB do Módulo Injetor

3) Selecione a interface de escrita da memória

Várias interfaces estão presentes para acesso ao banco de registro interno. Você pode usar todas elas simultaneamente para comandos de leitura, mas apenas uma pode ser utilizada por vez (e deve ser selecionada) para executar comandos de gravação. O interruptor "USERBIT2" seleciona a interface de gravação da memória. USERBIT2 = '0' significa" HDMI_I2C ativo"; USERBIT2 = '1' significa "DEBUG_I2C ativo". Para operar através do Módulo de Controle você deve selecionar "HDMI_I2C ativo".

4) Inicialize o setup *MiniDAQ*

Após um ciclo de energia os dispositivos que compõem seu sistema MiniDAQ (MiniDAQ, TELL40, CCPC, etc...) podem ter iniciado sem a configuração adequada. Consulte as instruções específicas para configurá-lo corretamente.

5) Inicialize o *GBT* do módulo de controle

Se o chip GBT no Módulo de Controle não possuir os "fusíveis queimados" você também deve carregar um mapa de registradores após cada ciclo de energia. O adaptador *I²C-USB* desenvolvido para operação autônoma do Módulo Injetor pode ser usado, mas será necessário incluir um deslocador de nível entre o dongle (3v3) e o Módulo de Controle (1v5). O conjunto de 366 registros que deve ser gravado (0x0000 a 0x016D) segue abaixo:

```
0000: 15 15 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 04 04 04 04 04 04 04 04 FF 01 7F 28 00 15 15 15
0020: 66 00 0D 42 00 0F 04 08 00 20 00 00 00 00 15 15
0030: 15 00 07 00 3F 00 00 00 00 00 00 00 00 00 00 00
0040: DD 0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050: 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00
0060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 15
00A0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0: 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00
00C0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00D0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00F0: 00 00 3F 3F 38 00 00 00 07 00 00 00 00 00 71 11
0100: FF 01 11 FF 01 11 FF 01 11 FF 01 FF FF 99 99 99
0110: 99 20 00 00 00 00 00 00 00 15 00 20 00 00 00 00
0120: 00 00 00 00 00 FF FF FF 40 40 40 2A 2A 2A 00 00
0130: FF FF FF 40 40 40 2A 2A 2A 7B 7B 7B AA 0A 07 00
0140: FF FF FF FF FF FF FF 00 00 00 00 00 01 11 FF 01
0150: 11 FF 01 11 FF 01 11 FF 01 FF 00 01 11 FF 01 11
0160: FF 01 11 FF 01 11 FF 01 FF FF 00 00 00 AA -- --
```

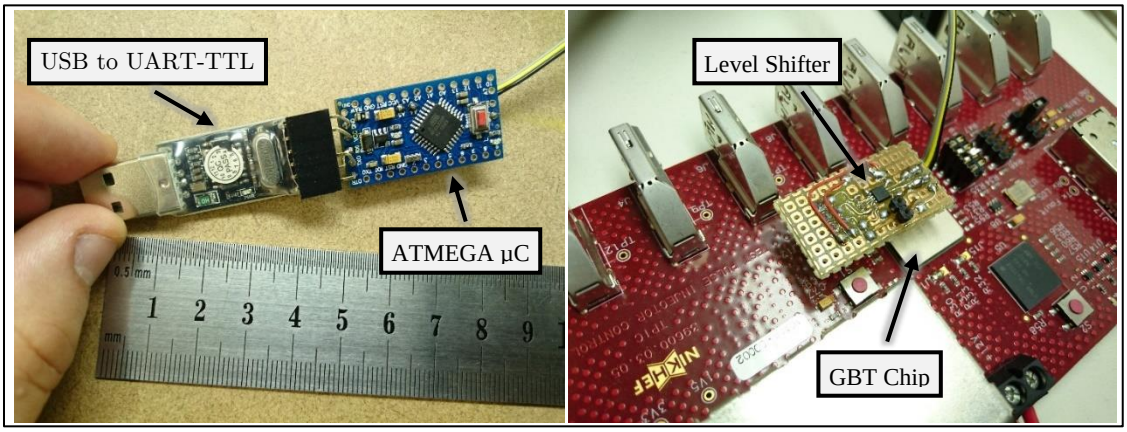


Figura 6 – Adaptador I2C, à esquerda, e deslocador de nível, à direita

6) Inicialize o painel de controle no *WinCC*

Após um ciclo de energia o Módulo Injetor inicializa com o valor 0x00 em quase todos os registradores internos, e isto significa "sem sinal em todas as saídas". Então, para ver um sinal você deve escrever em alguns registradores. Use os botões de controle, controles deslizantes e caixas presentes no painel de controle WinCC para ativar o canal desejado.

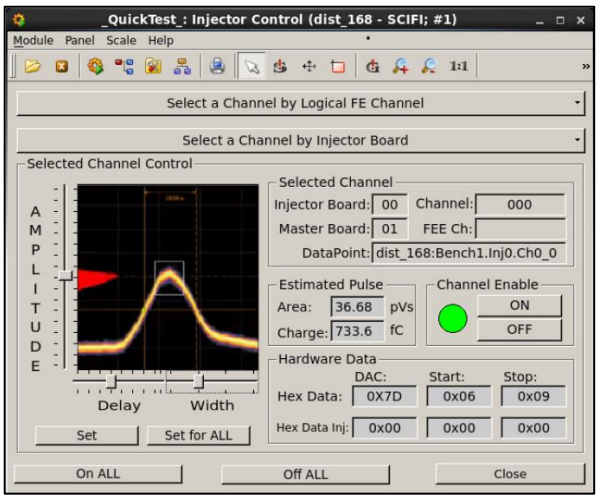


Figura 7 – Painel de controle no WinCC

Apêndice B

Diagramas Esquemáticos do Injetor de Carga

Índice de Diagramas deste Apêndice:

Diagrama 1 – Injetor de Carga 256 Canais (Folha Principal do Projeto) 148
Diagrama 2 – Reguladores de Tensão dos Circuitos Analógicos 149
Diagrama 3 – Reguladores de Tensão dos Circuitos Digitais 150
Diagrama 4 – Circuitos Analógicos (Folha Principal) 151
Diagrama 5 – Circuitos Analógicos (Grupo de 16 canais) 152
Diagrama 6 – Circuito de Ganho e Compensação de Offset 153
Diagrama 7 – Circuito de Modulação de Amplitude e Buffer de Saída 154
Diagrama 8 - Circuitos Digitais (Folha Principal) 155
Diagrama 9 – 1º Banco de Entradas e Saídas do FPGA 156
Diagrama 10 – 2º Banco de Entradas e Saídas do FPGA 157
Diagrama 11 – 3º Banco de Entradas e Saídas do FPGA 158
Diagrama 12 – 4º Banco de Entradas e Saídas do FPGA 159
Diagrama 13 – 5º Banco de Entradas e Saídas do FPGA 160
Diagrama 14 – 6º Banco de Entradas e Saídas do FPGA 161
Diagrama 15 – 7º Banco de Entradas e Saídas do FPGA 162
Diagrama 16 – 8º Banco de Entradas e Saídas do FPGA 163
Diagrama 17 – Conexões de Clock, Configuração e Alimentação do FPGA 164
Diagrama 18 – De-serializadores dos Comandos de Chip-Select dos DAC's 165
Diagrama 19 – Buffers dos Sinais de Controle dos DAC's 166
Diagrama 20 – Interfaces Multi Propósito e de Usuário 167
Diagrama 21 – Partes Mecânicas e Especificações da Placa de Circuito Impresso ... 168

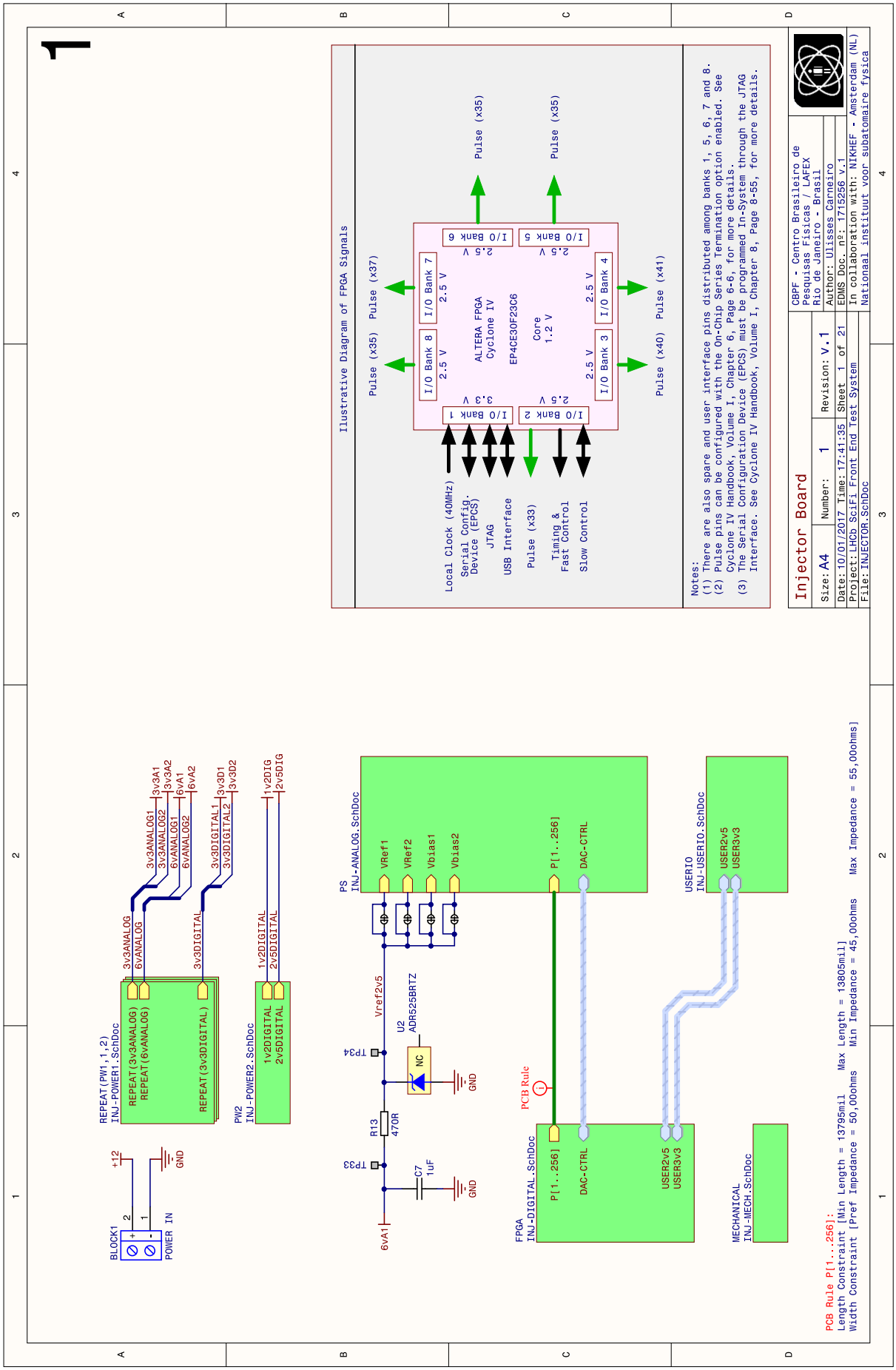


Diagrama 1 – Injetor de Carga 256 Canais (Folha Principal do Projeto)

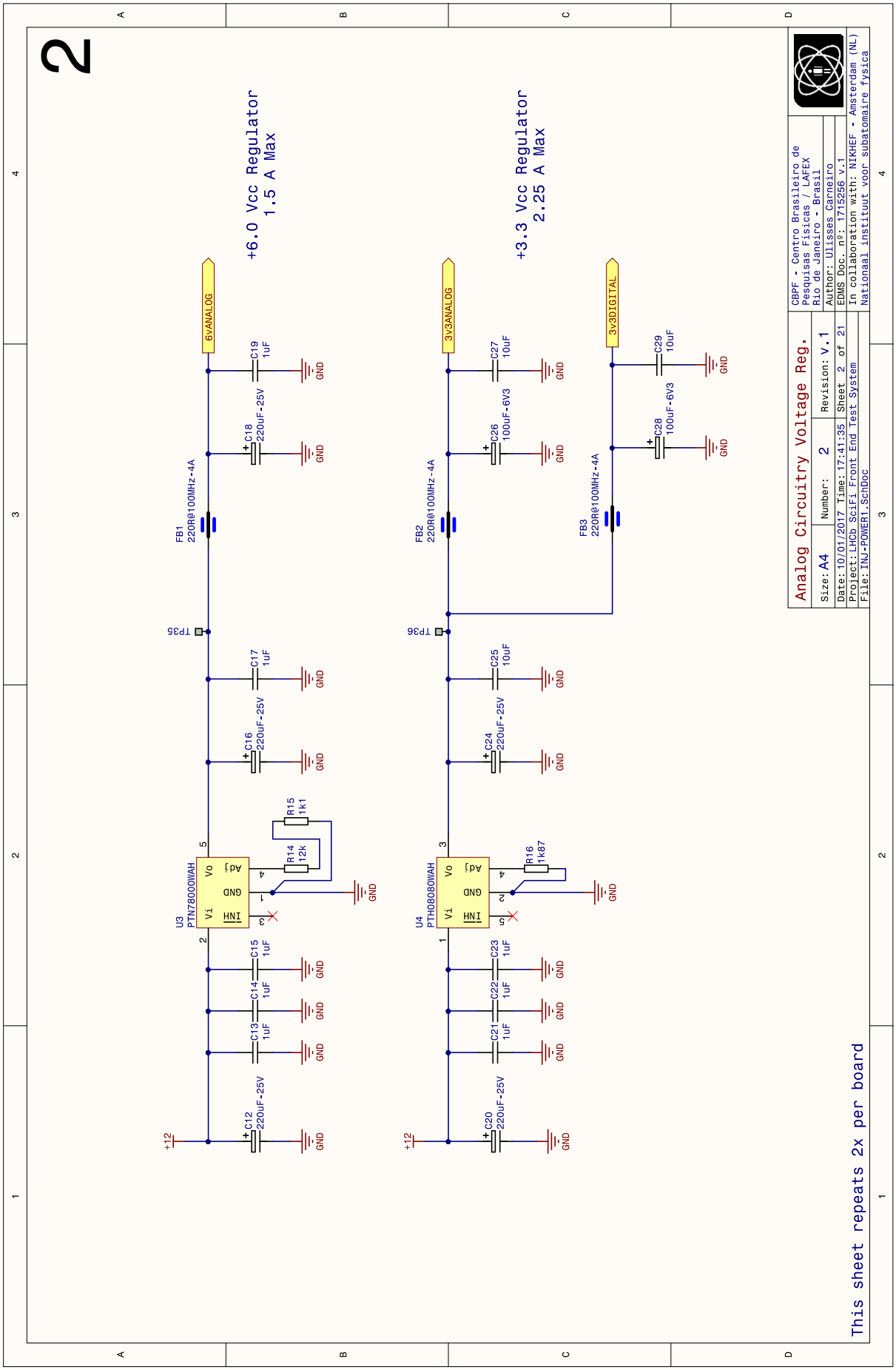


Diagrama 2 – Reguladores de Tensão dos Circuitos Analógicos

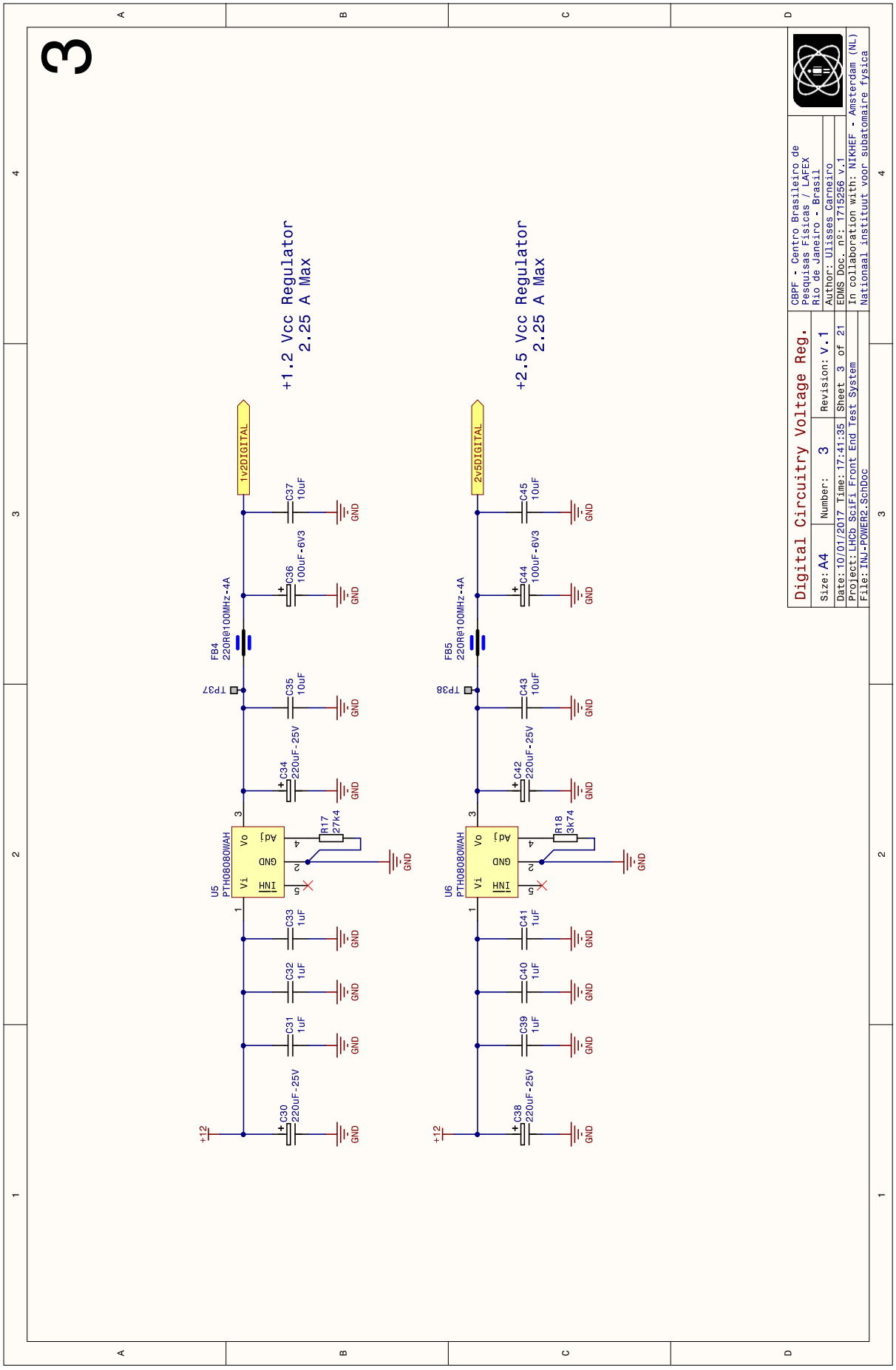


Diagrama 3 – Reguladores de Tensão dos Circuitos Digitais

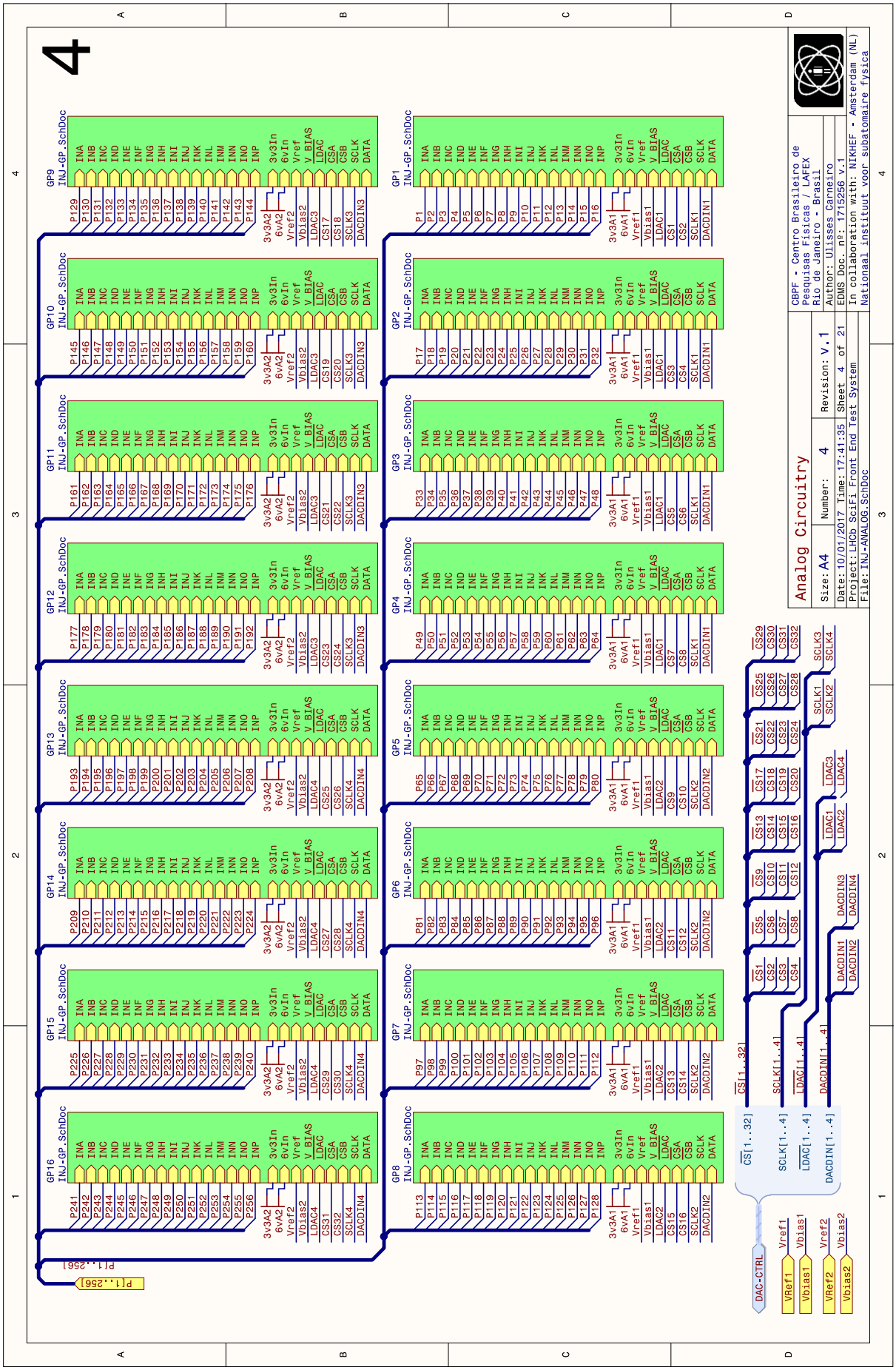


Diagrama 4 – Circuitos Analógicos (Folha Principal)

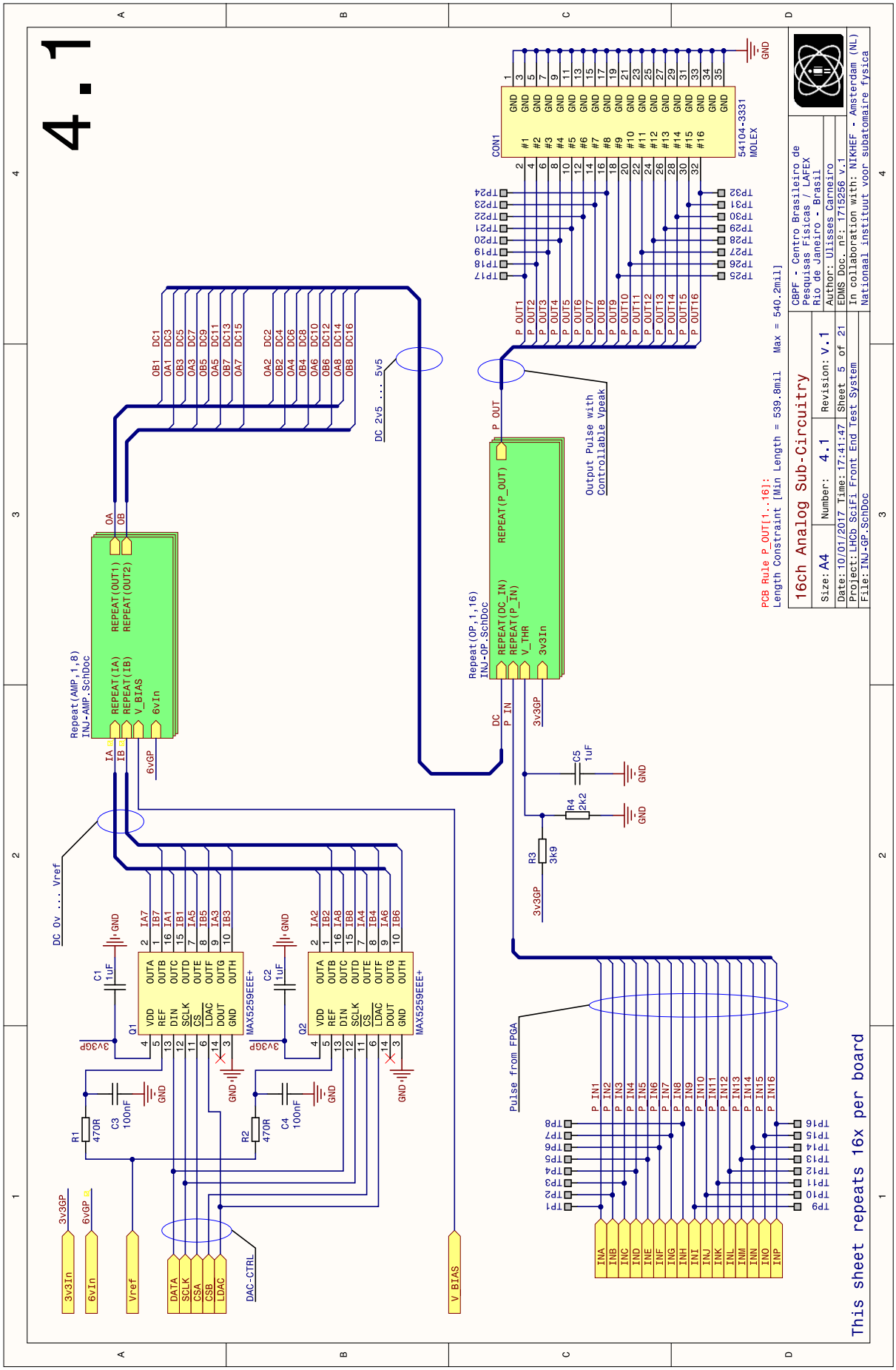


Diagrama 5 – Circuitos Analógicos (Grupo de 16 canais)

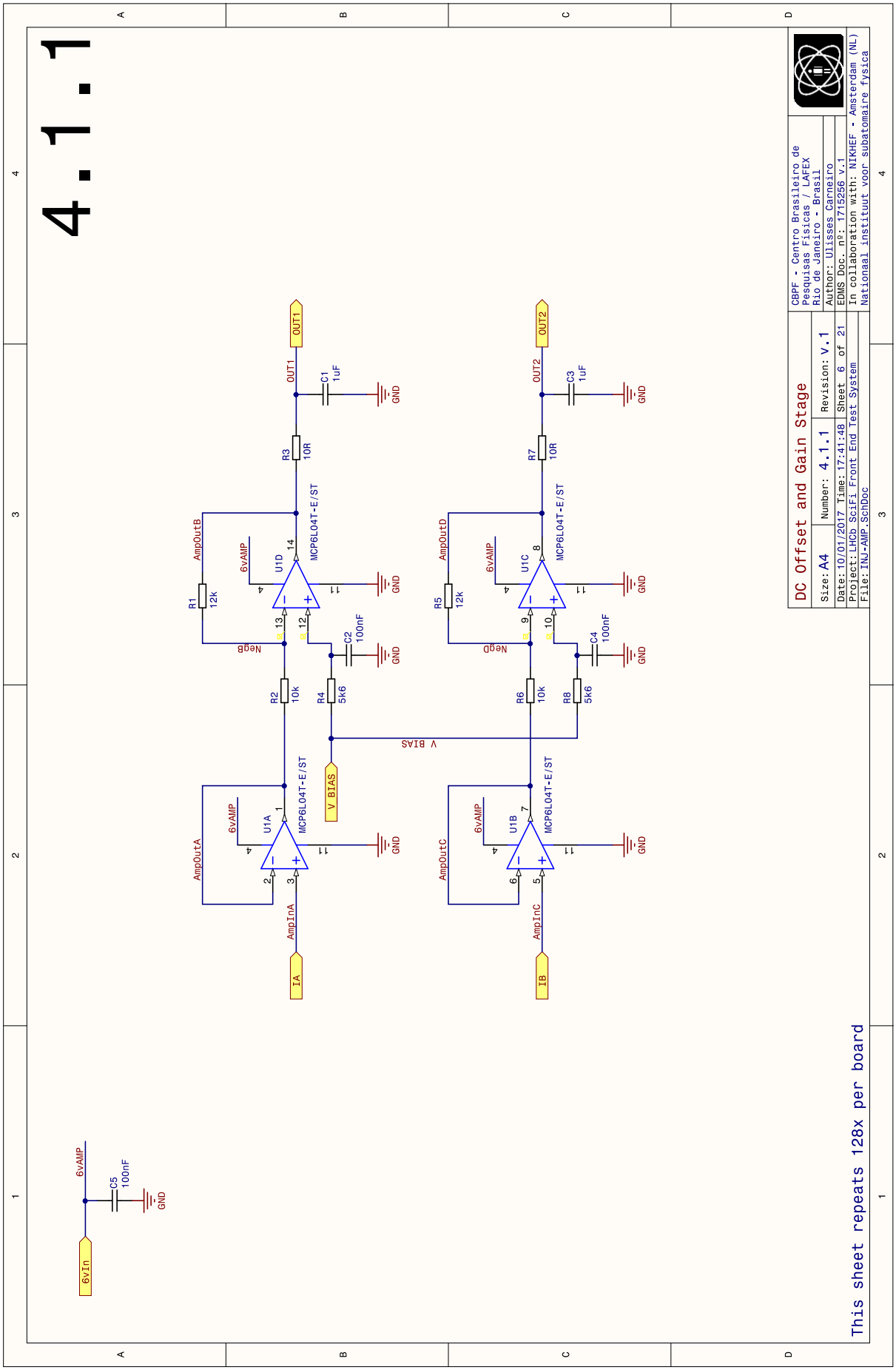


Diagrama 6 – Circuito de Ganho e Compensação de Offset

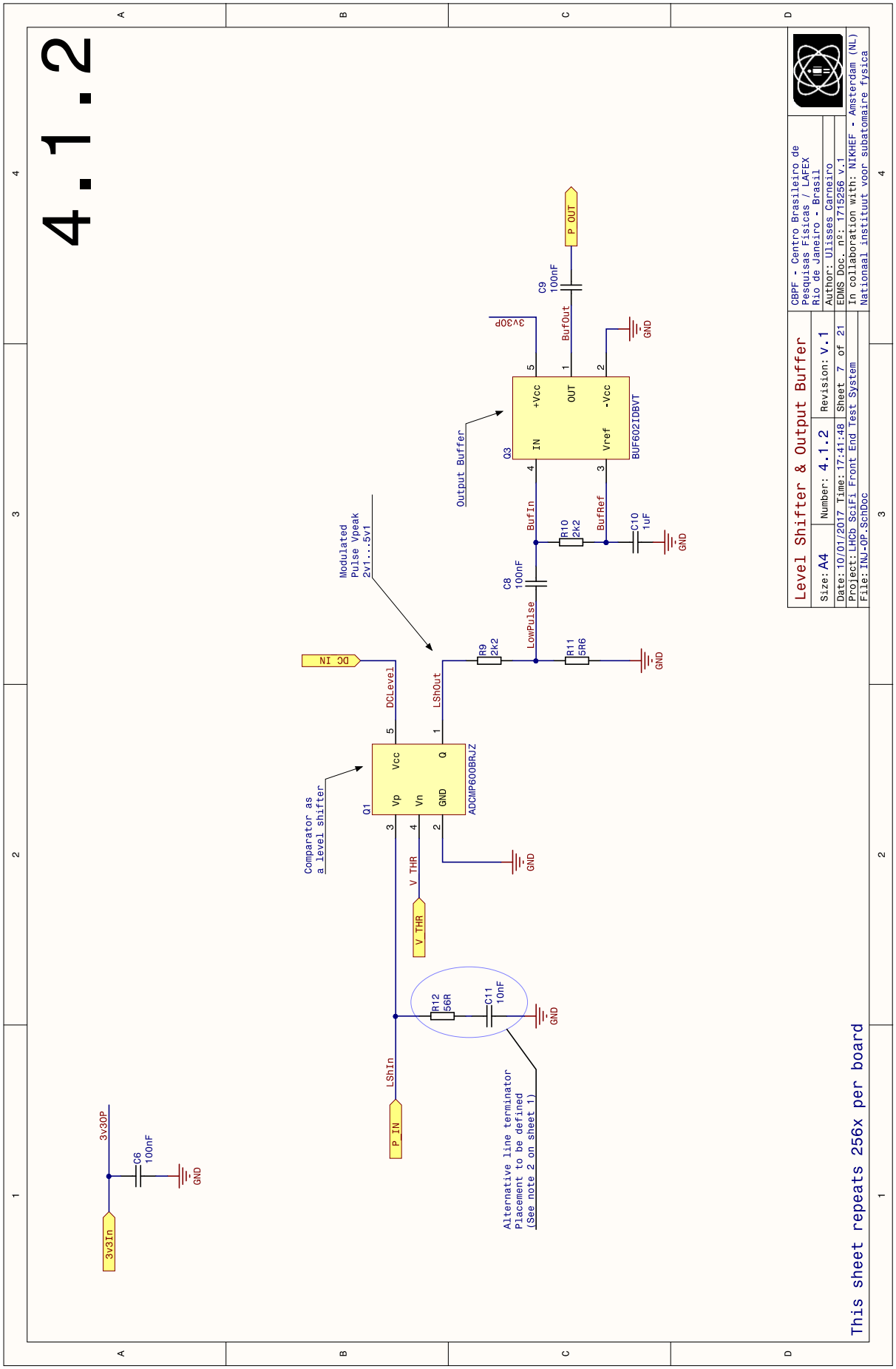


Diagrama 7 – Circuito de Modulação de Amplitude e *Buffer* de Saída

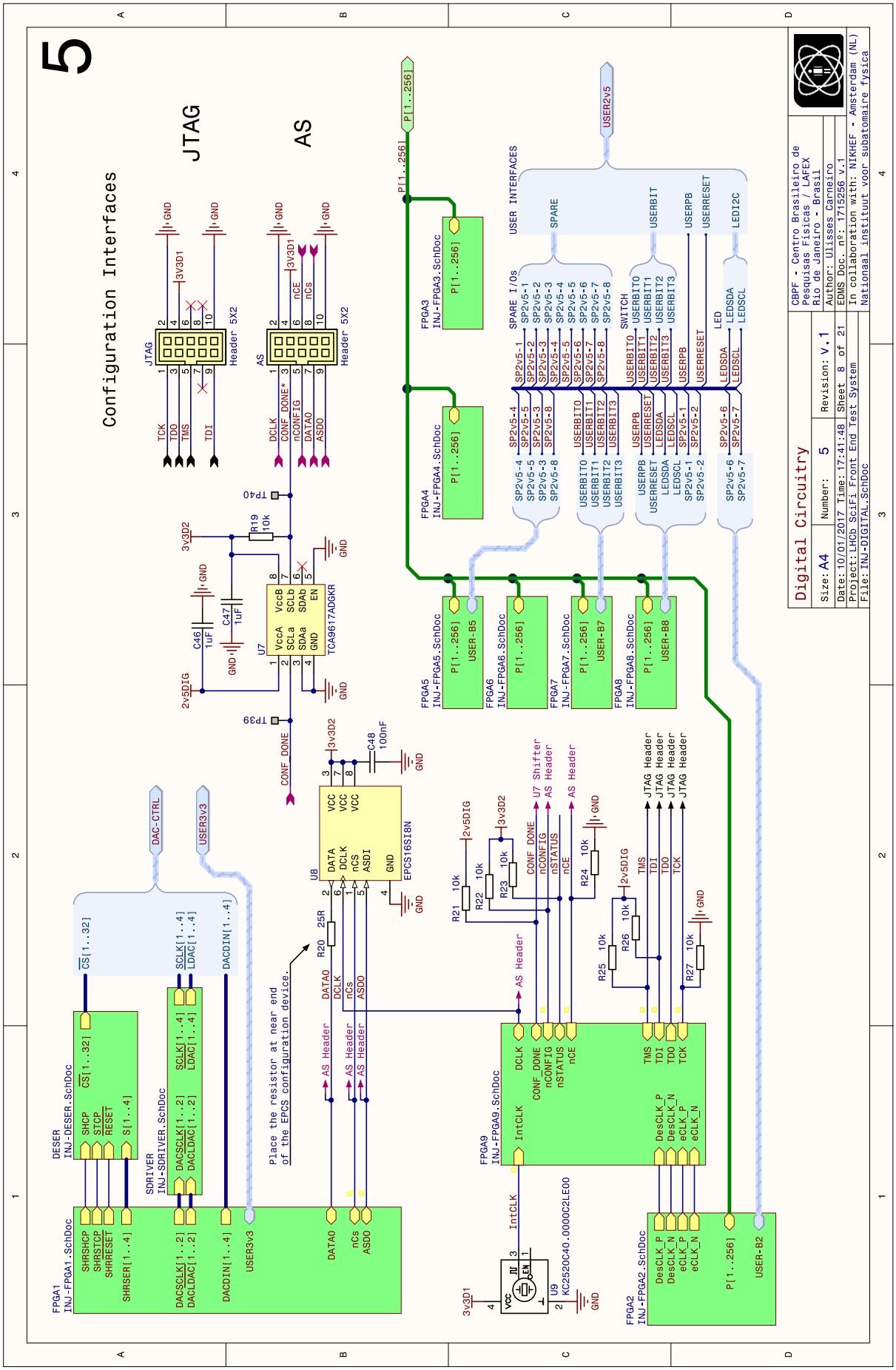


Diagrama 8 - Circuitos Digitais (Folha Principal)

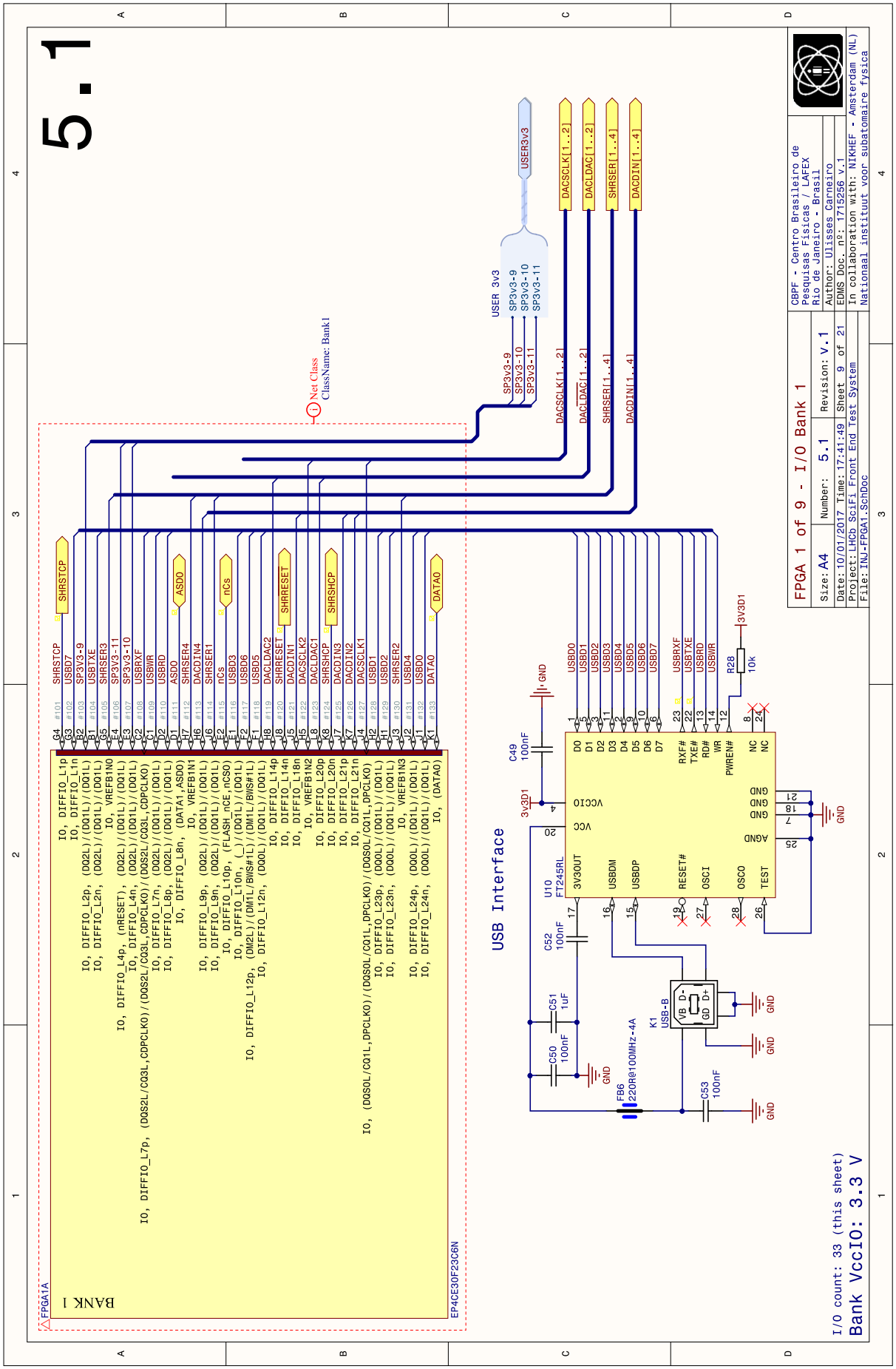


Diagrama 9 – 1º Banco de Entradas e Saídas do FPGA

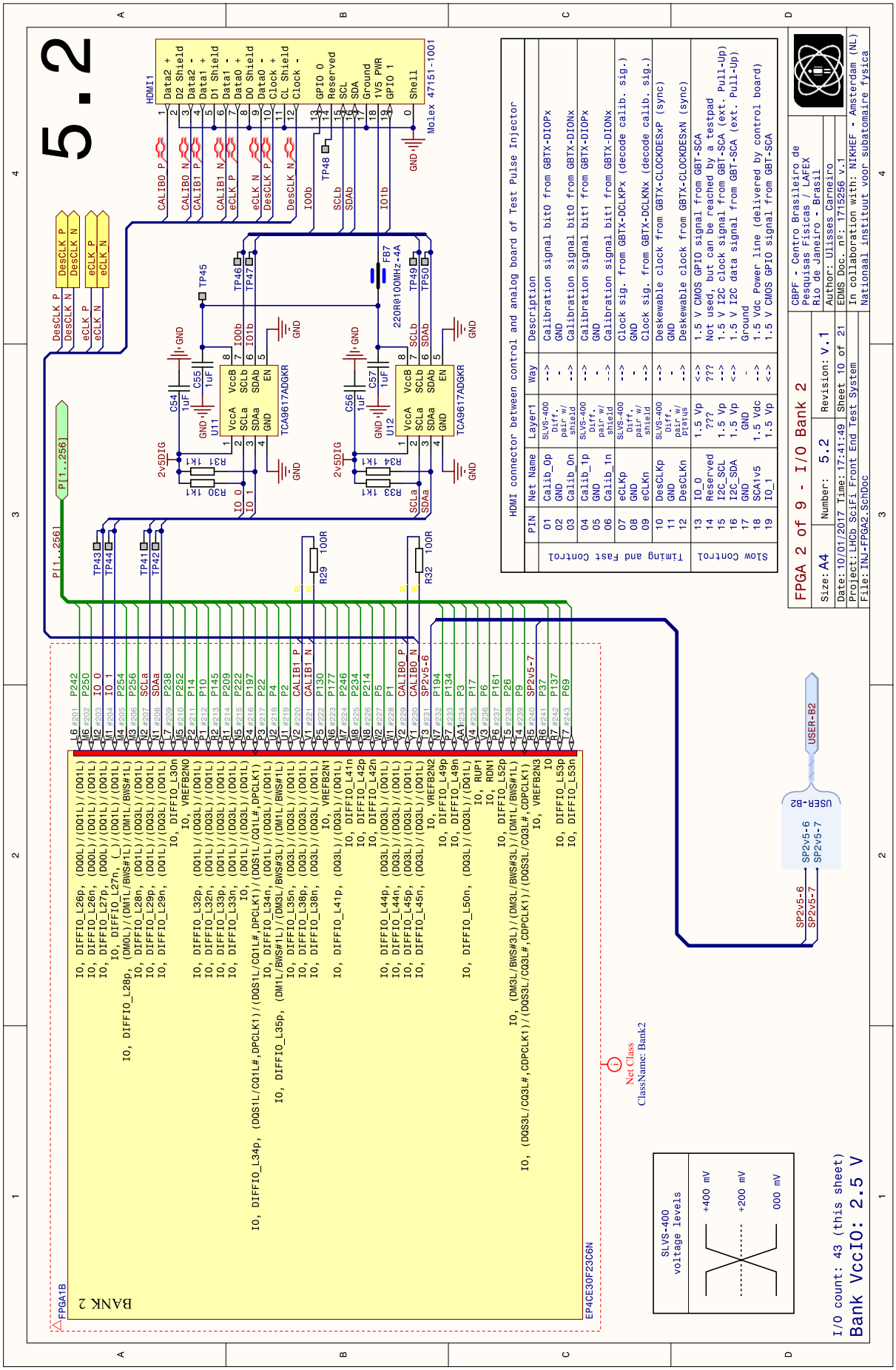


Diagrama 10 – 2º Banco de Entradas e Saídas do FPGA

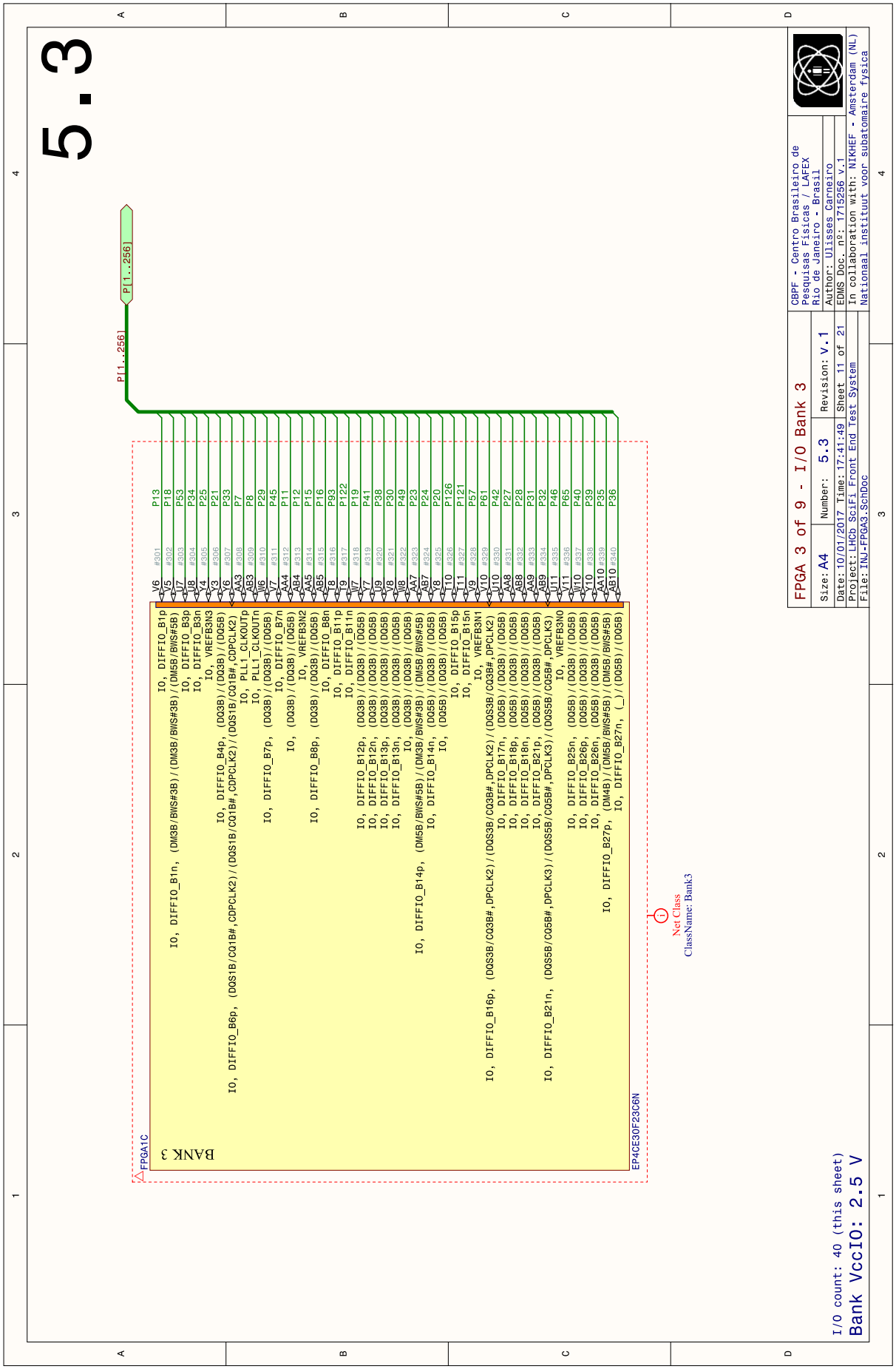


Diagrama 11 – 3º Banco de Entradas e Saídas do FPGA

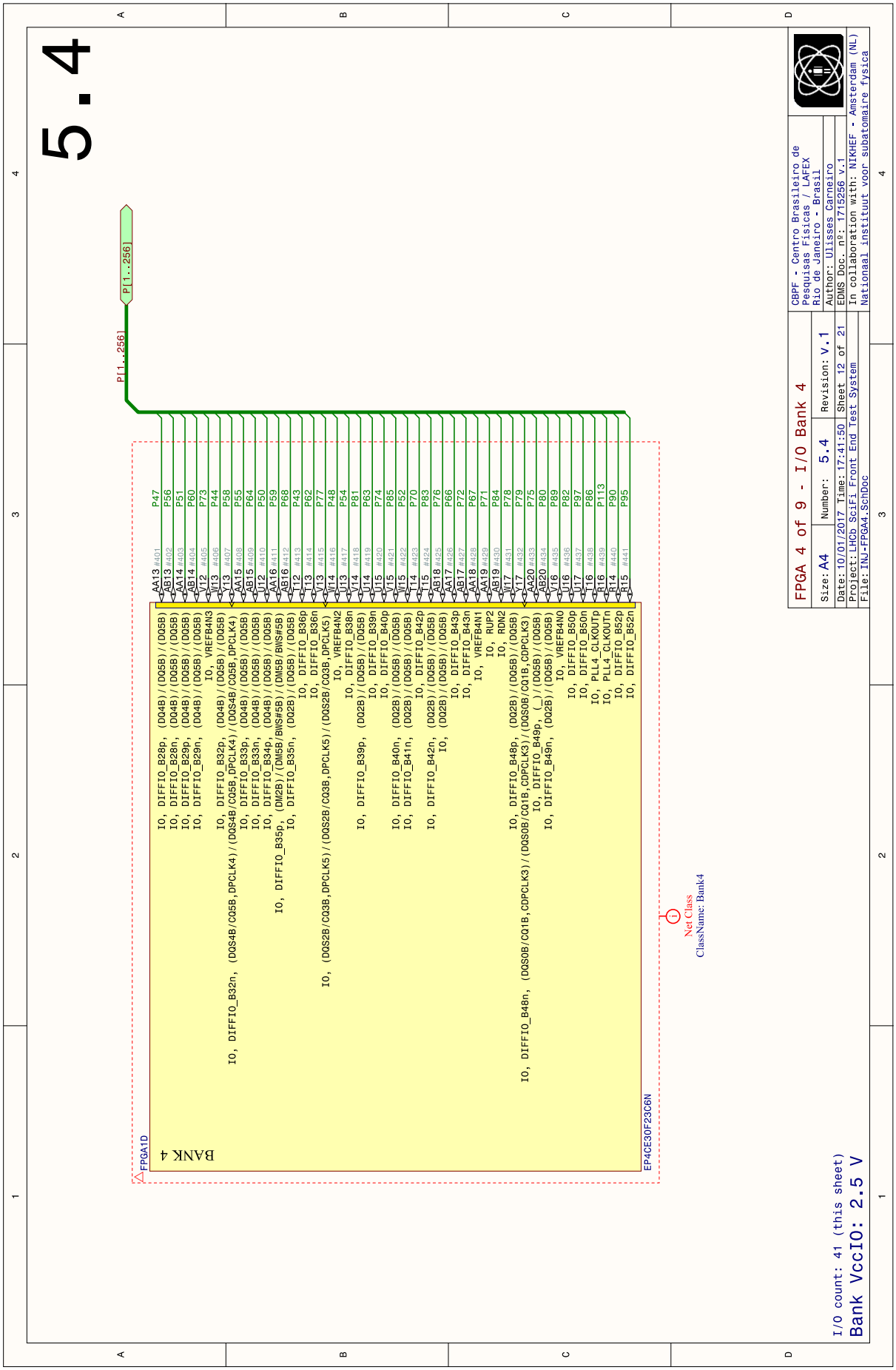


Diagrama 12 – 4º Banco de Entradas e Saídas do FPGA

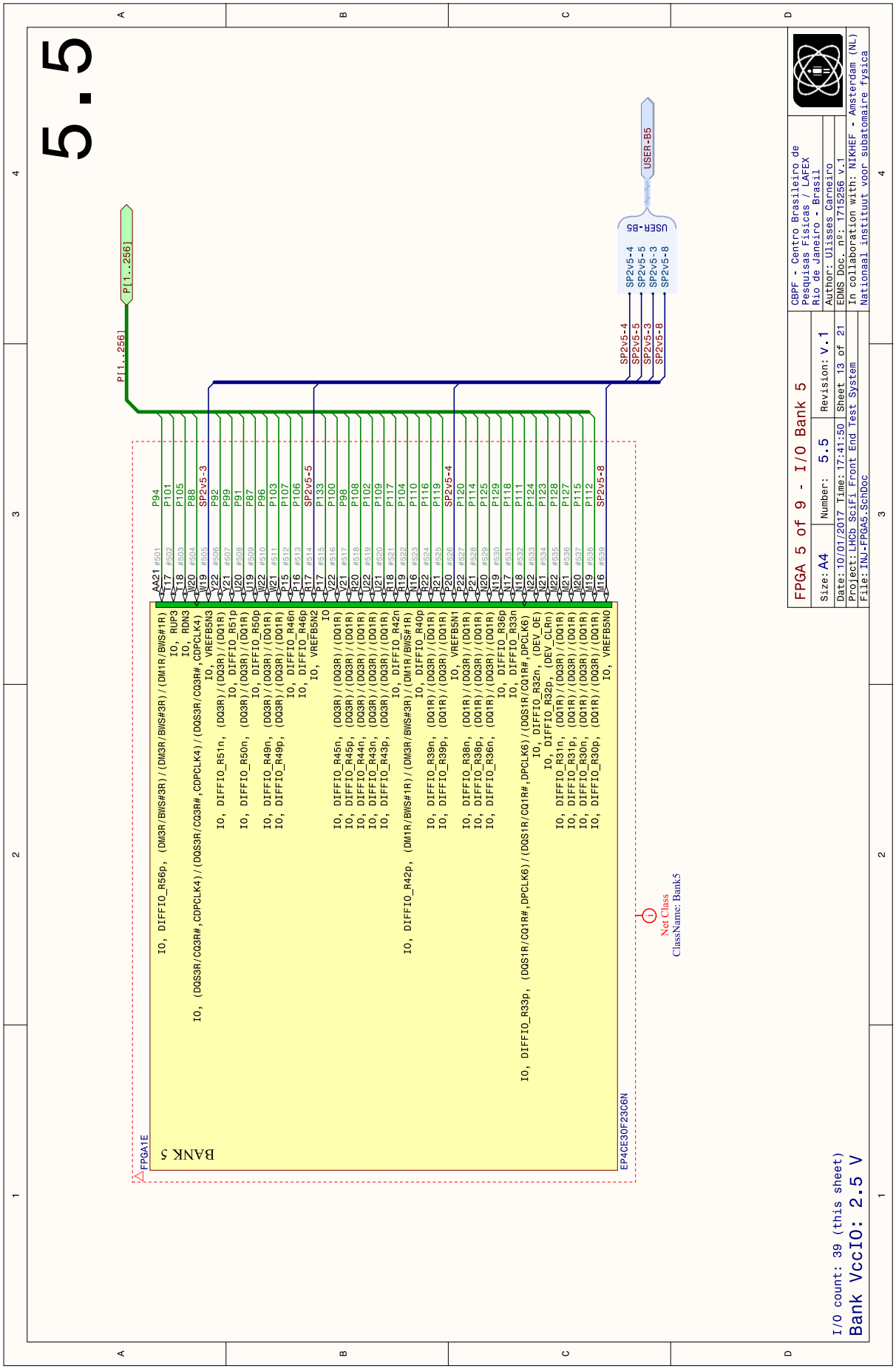


Diagrama 13 – 5º Banco de Entradas e Saídas do FPGA

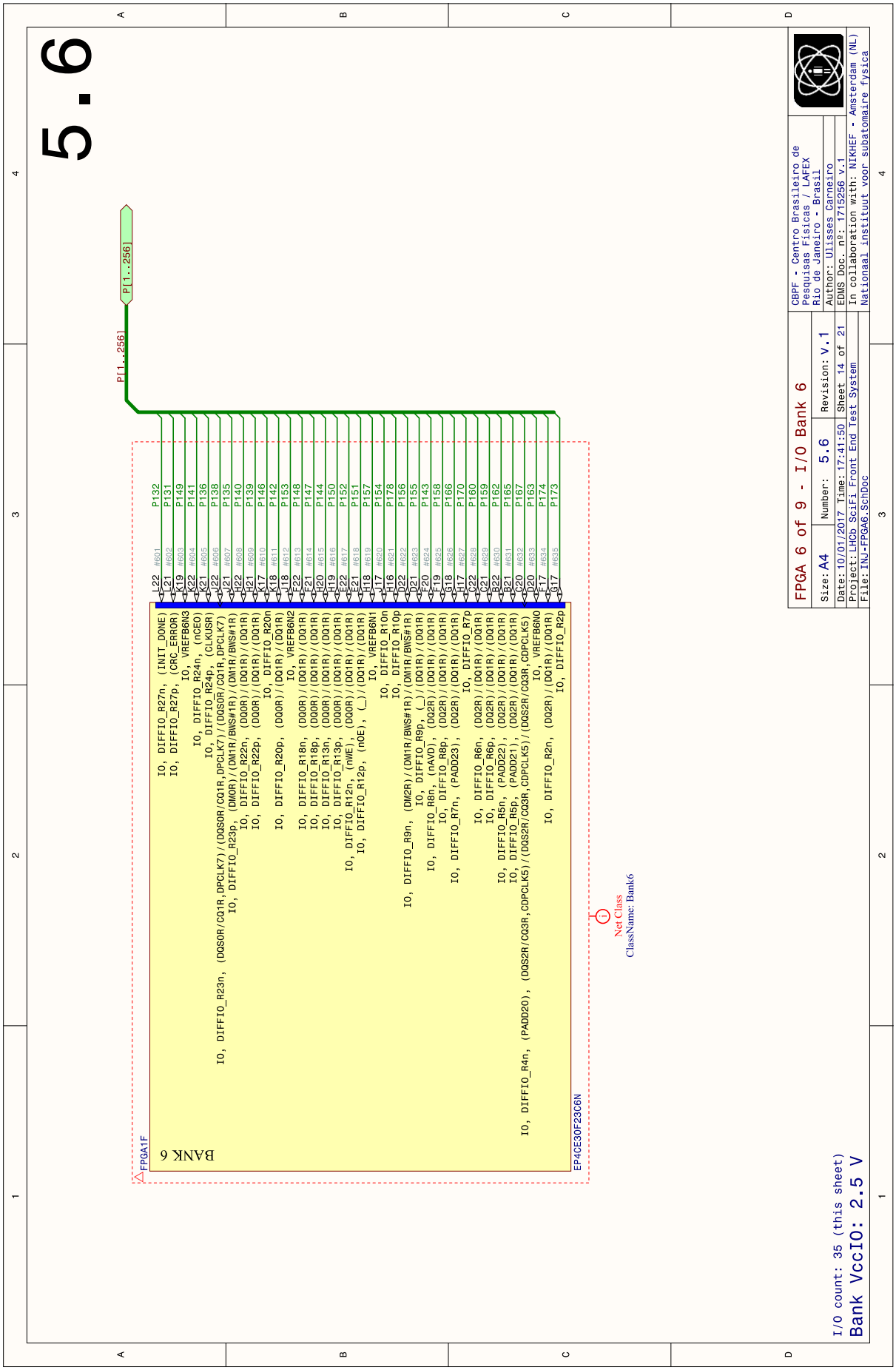


Diagrama 14 – 6º Banco de Entradas e Saídas do FPGA

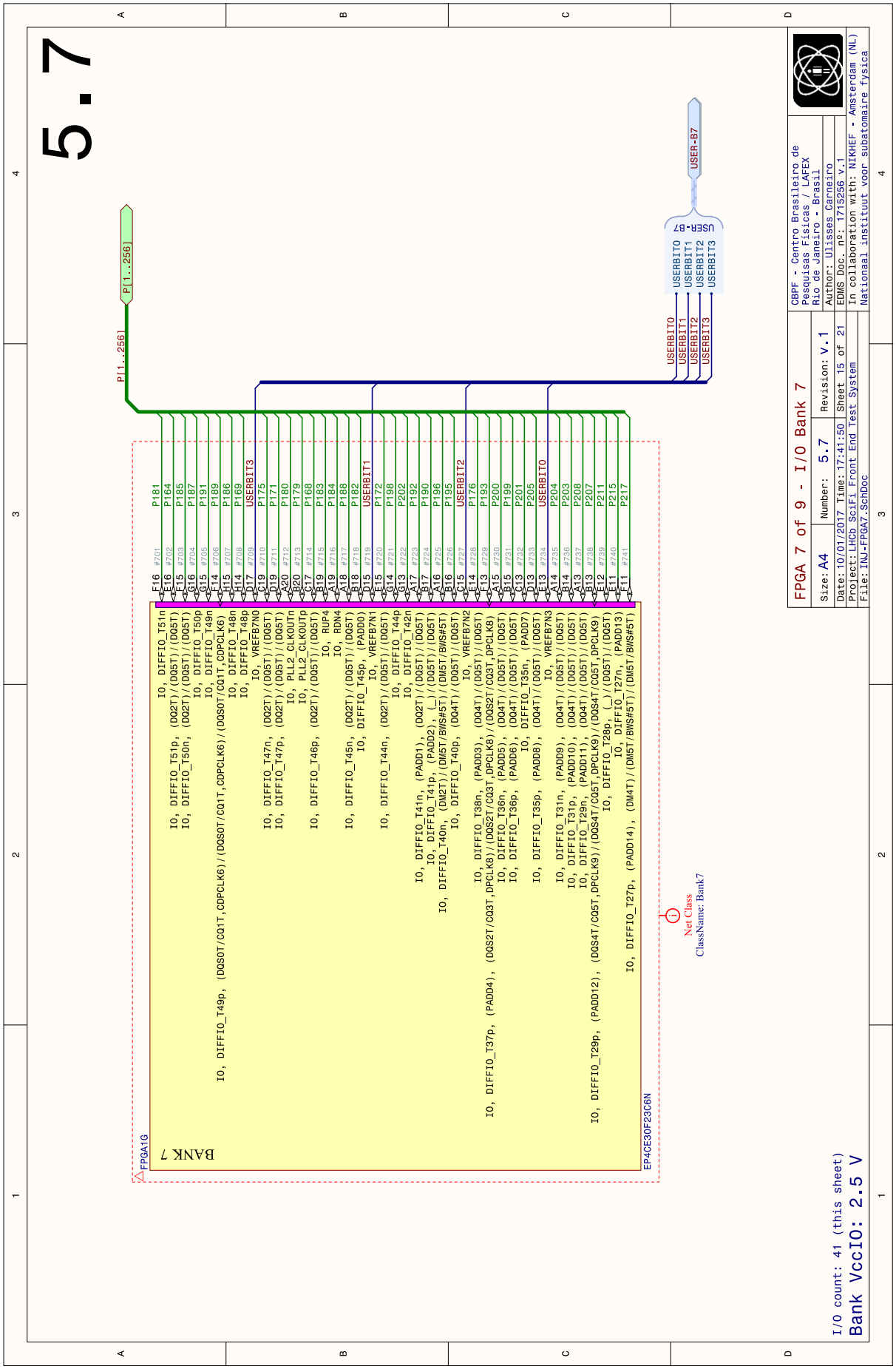


Diagrama 15 – 7º Banco de Entradas e Saídas do FPGA

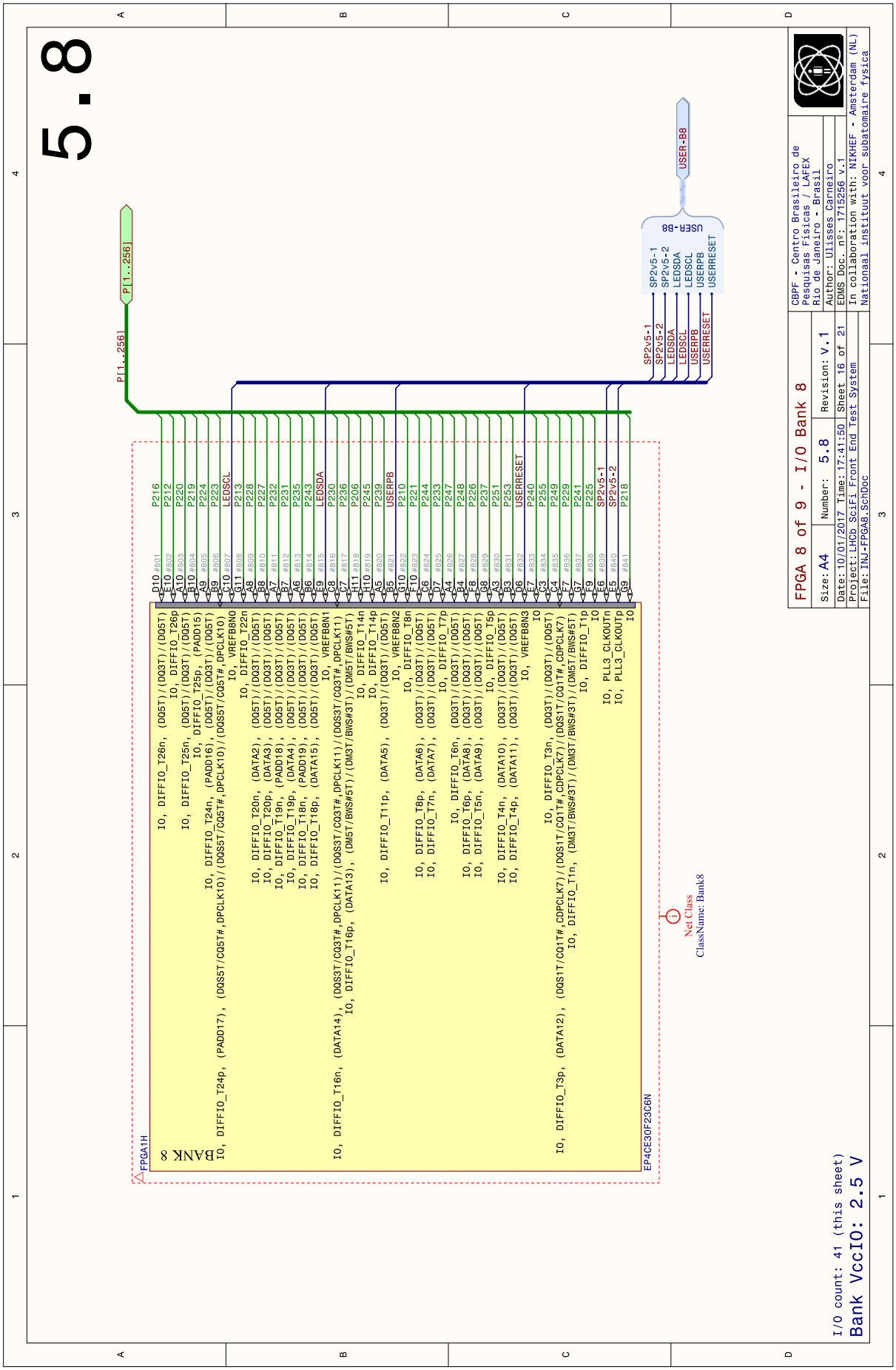


Diagrama 16 – 8º Banco de Entradas e Saídas do FPGA

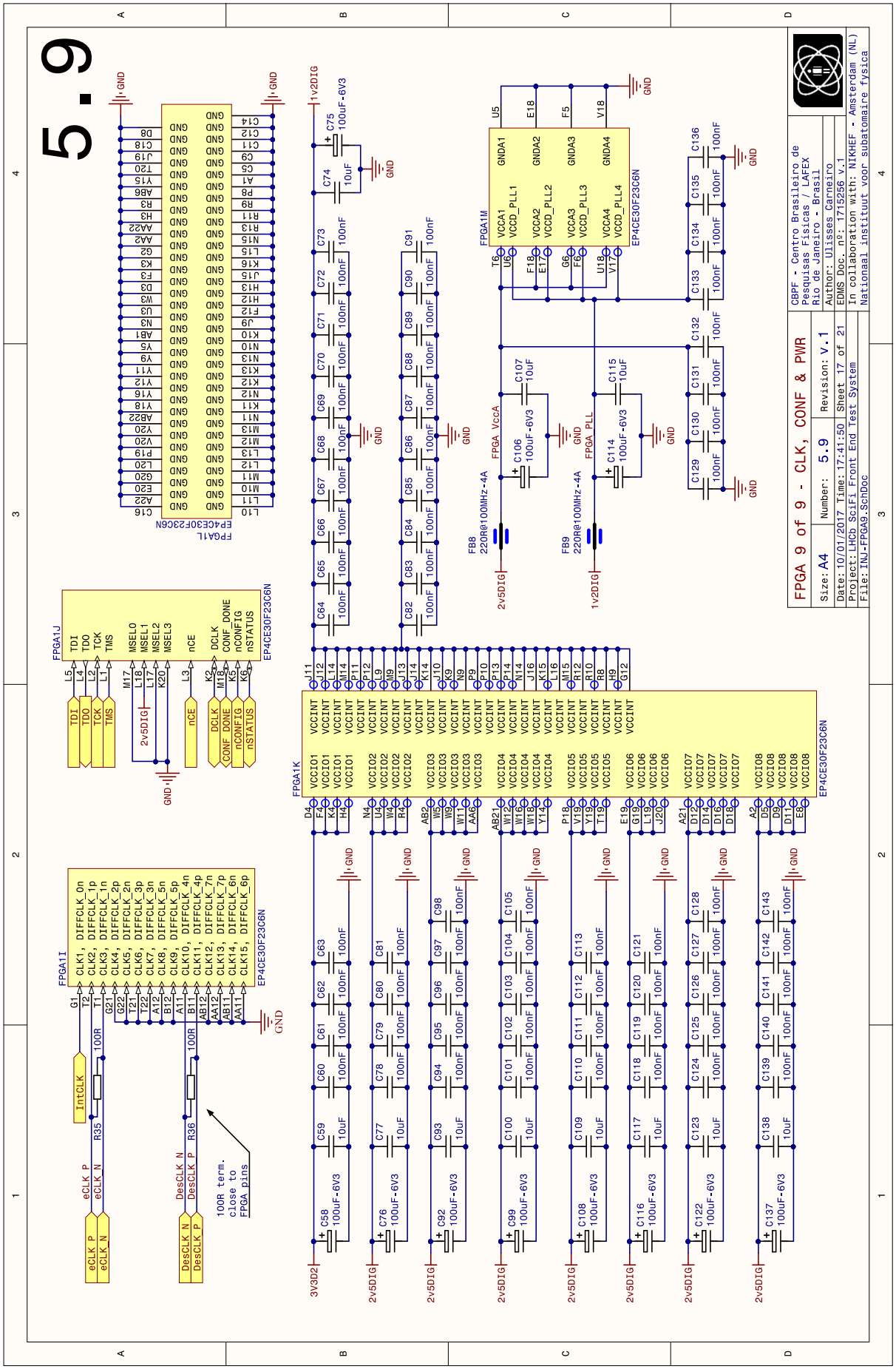


Diagrama 17 – Conexões de Clock, Configuração e Alimentação do FPGA

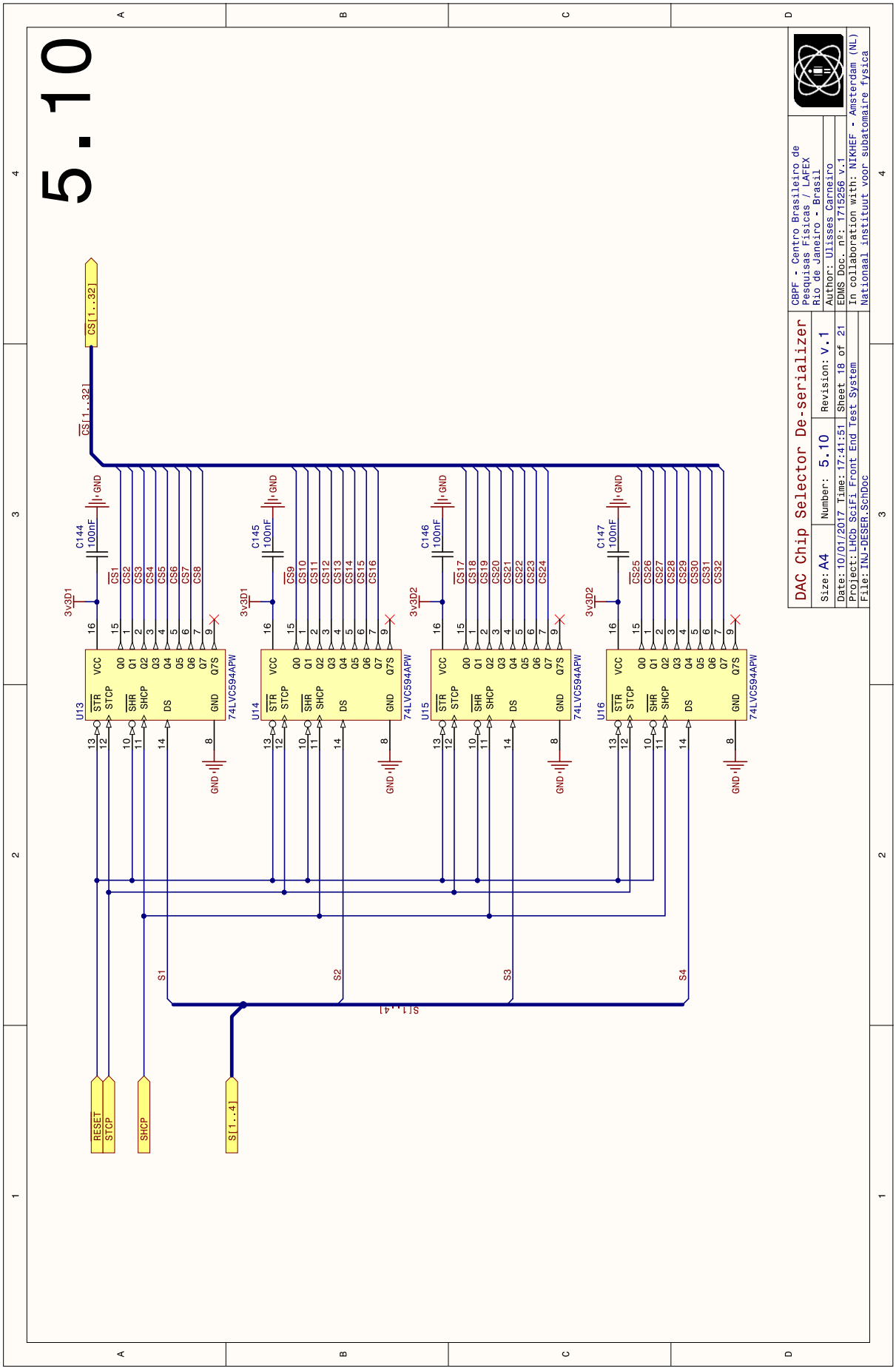


Diagrama 18 – De-serializadores dos Comandos de Chip-Select dos DAC's

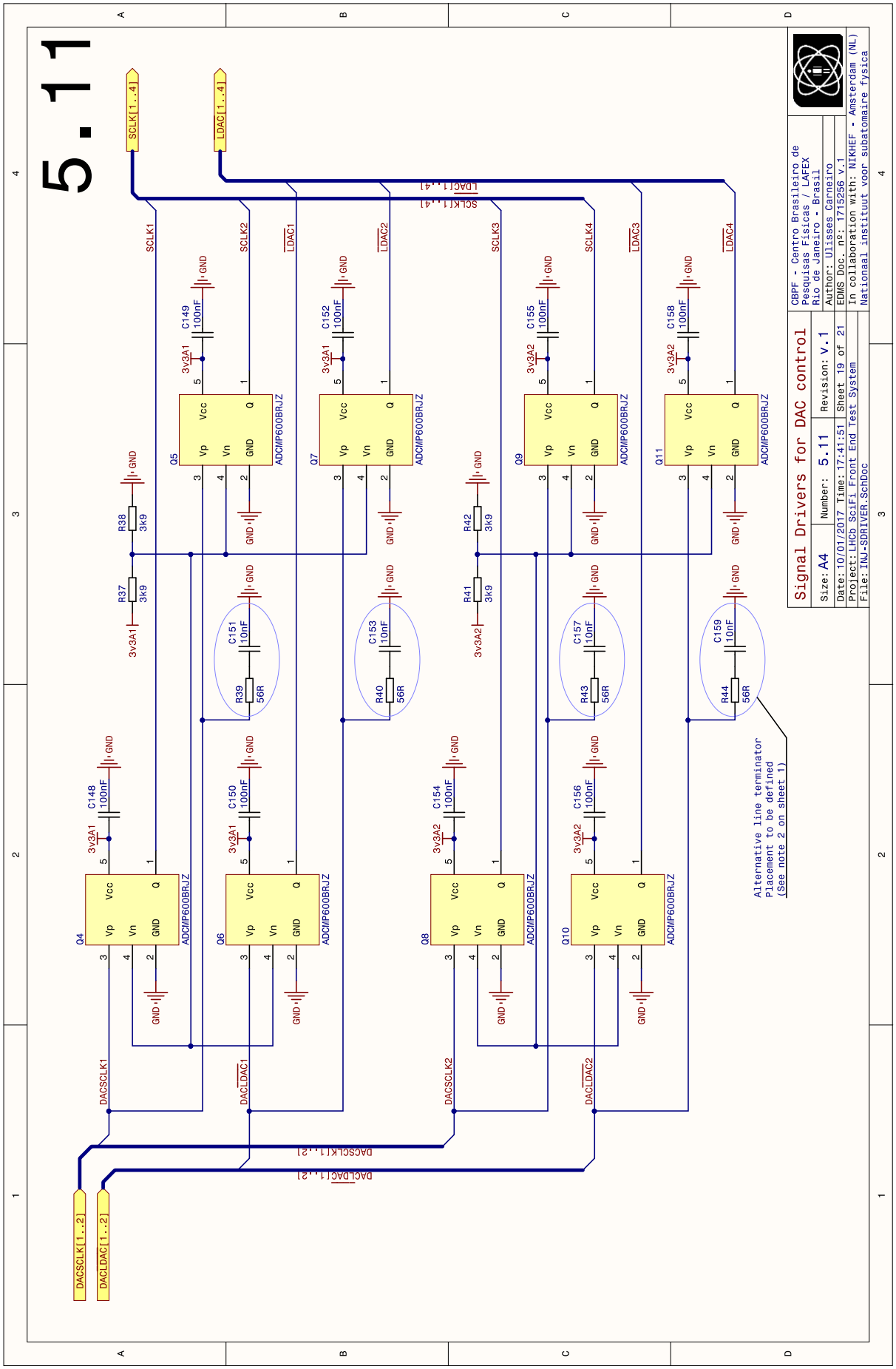


Diagrama 19 – *Buffers* dos Sinais de Controle dos DAC's

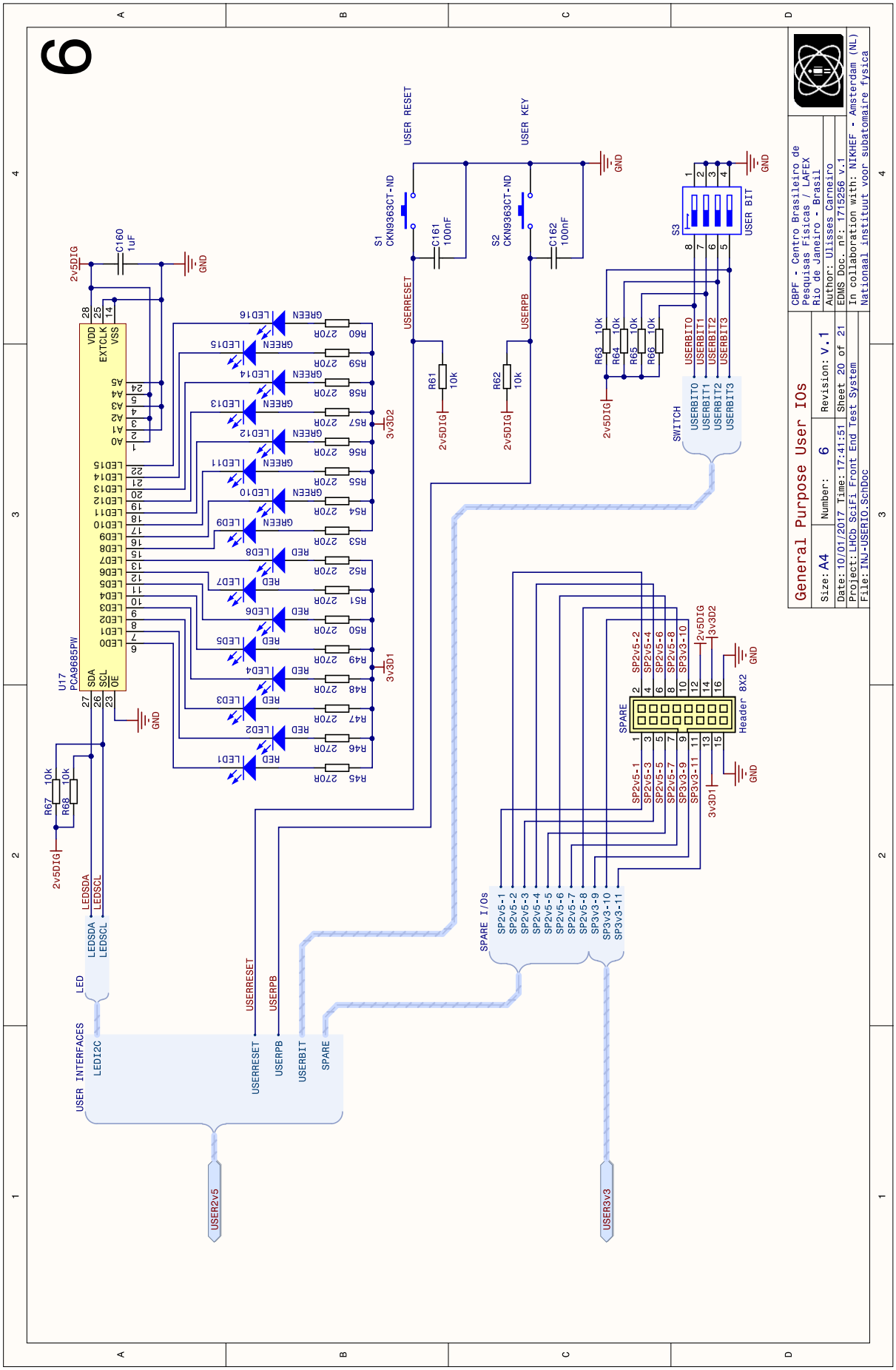


Diagrama 20 – Interfaces Multi Propósito e de Usuário

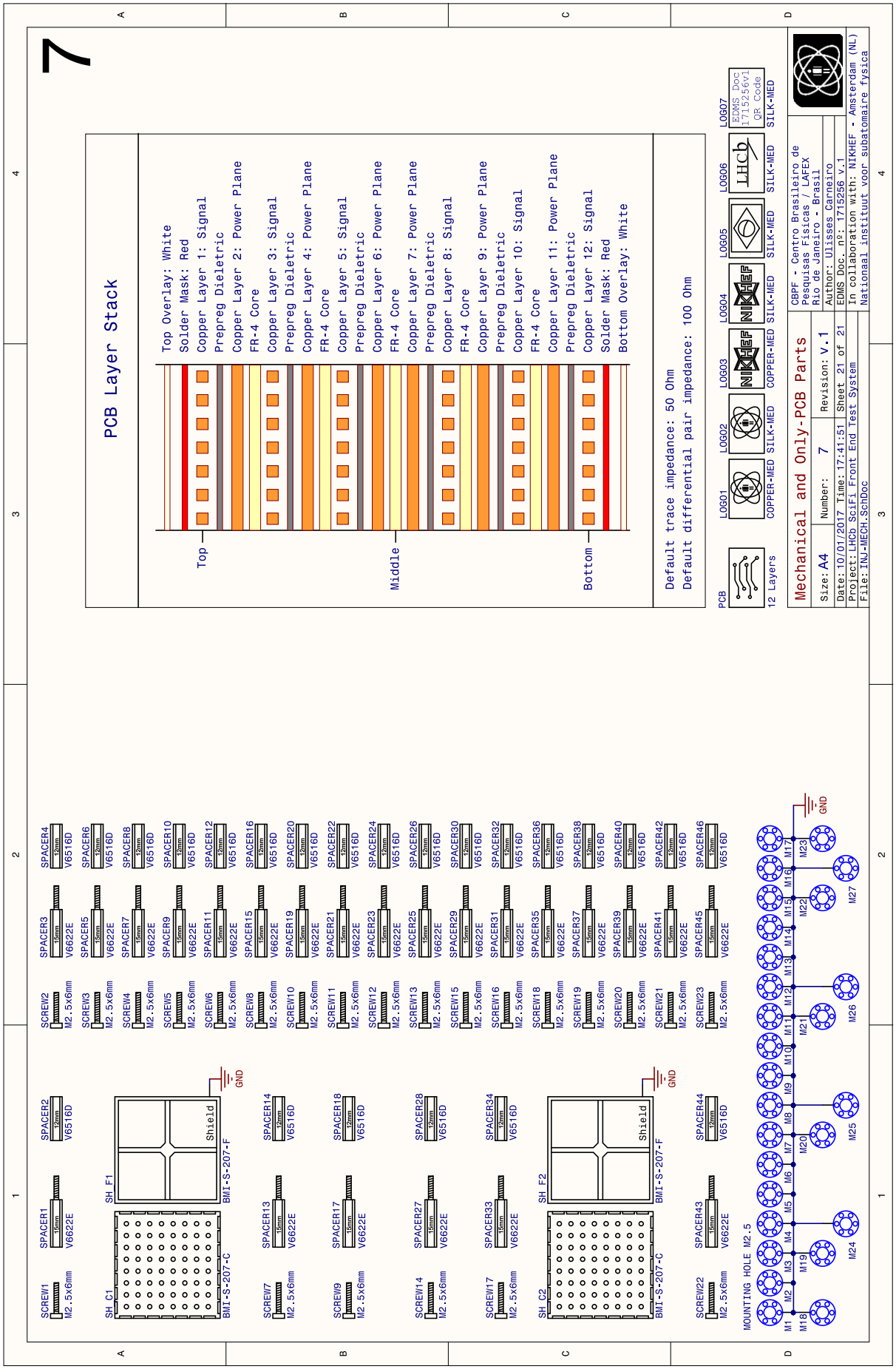


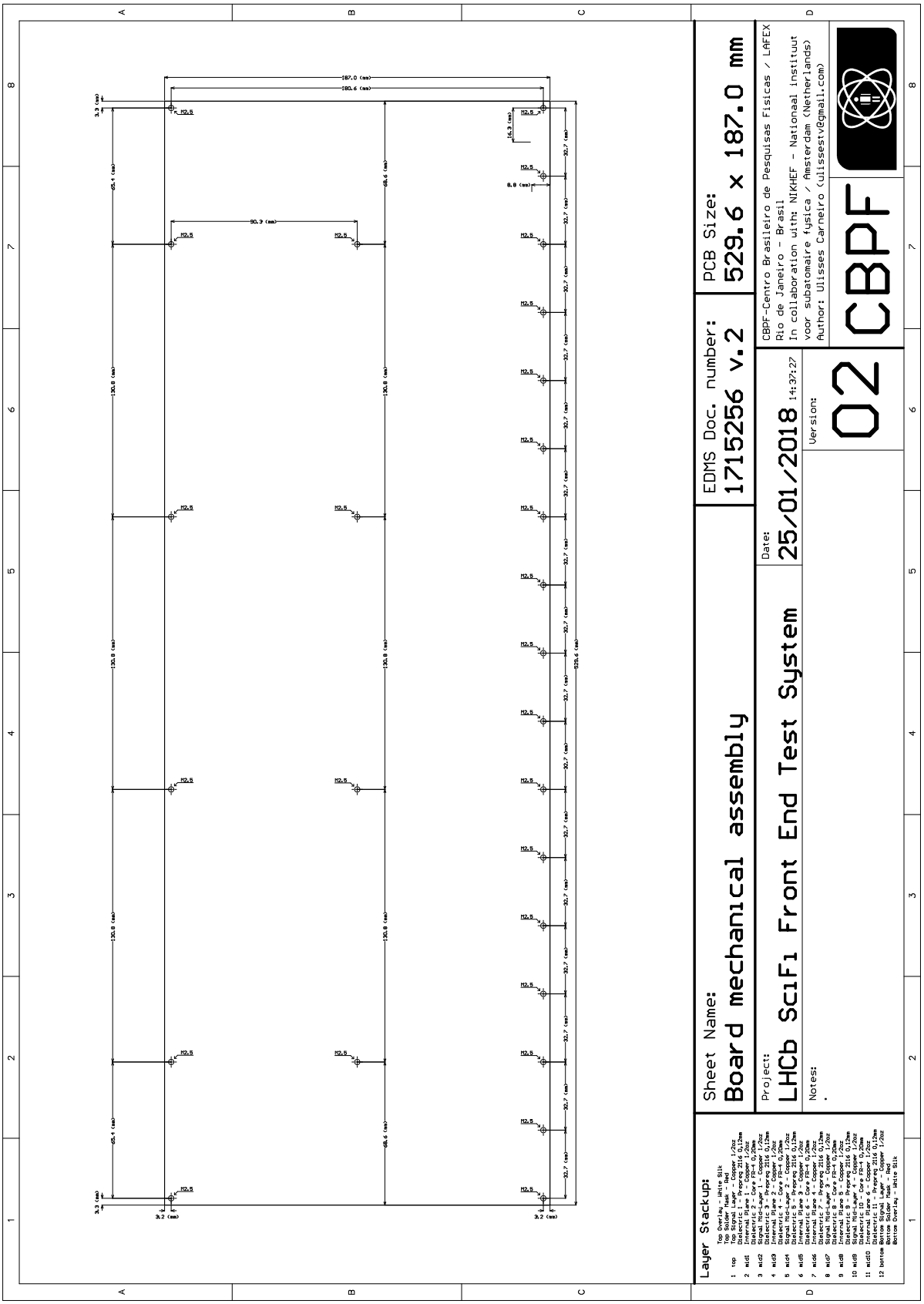
Diagrama 21 – Partes Mecânicas e Especificações da Placa de Circuito Impresso

Apêndice C

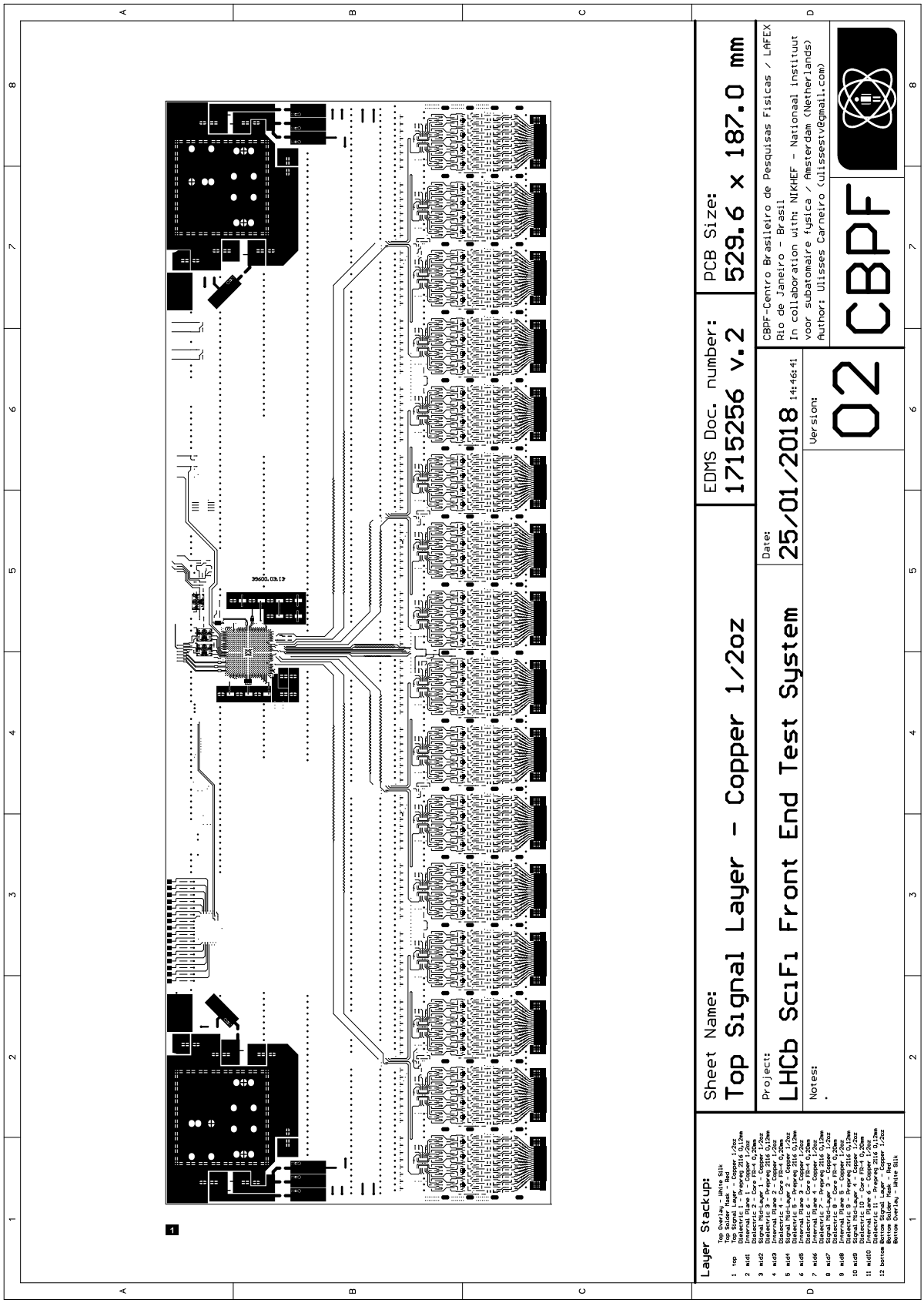
Desenhos da PCB do Módulo Injetor

Índice de Diagramas deste Apêndice:

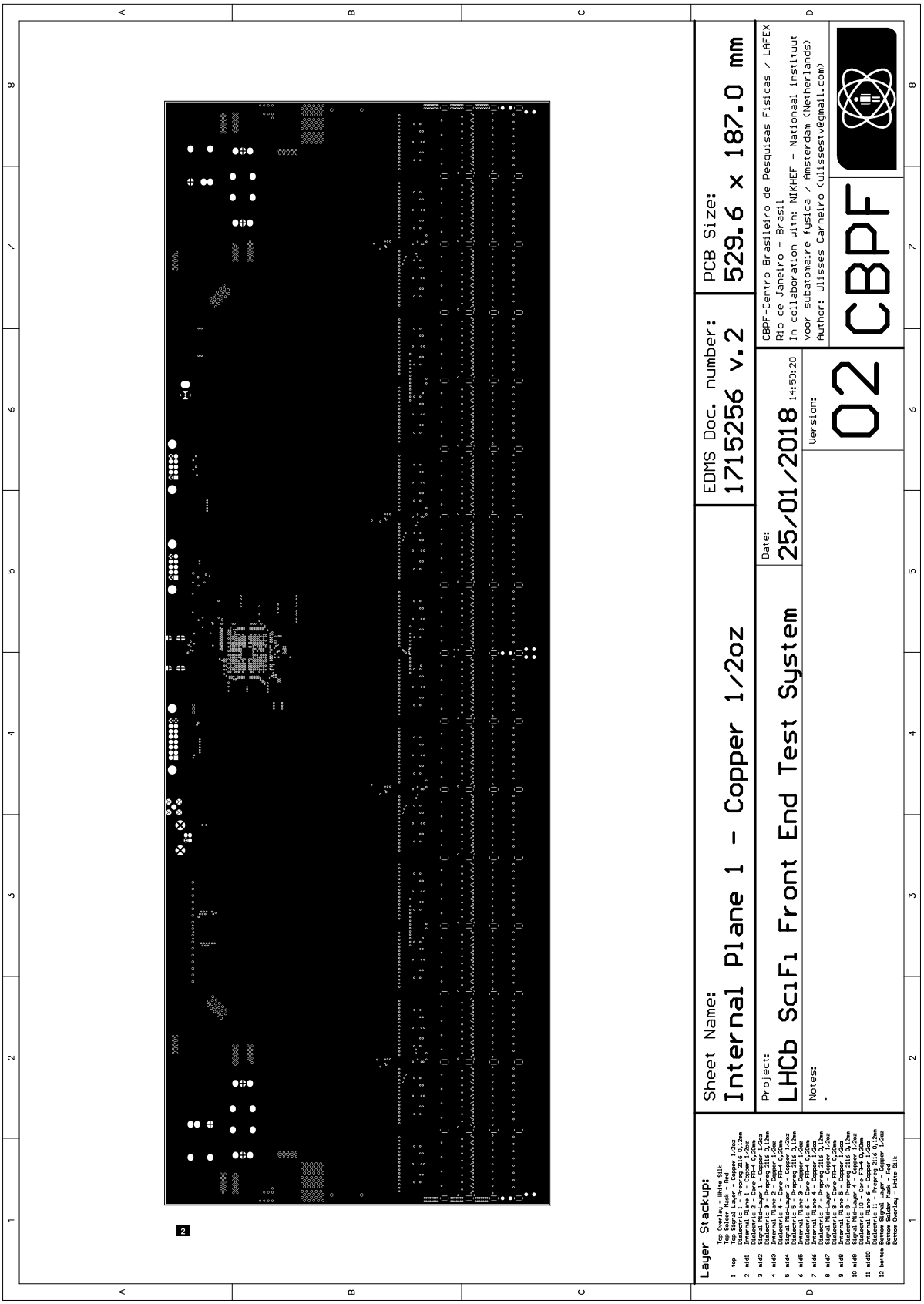
Desenho 1 – Dimensoões da Placa de Circuito Impresso 170
Desenho 2 – 1ª camada condutora: TOP 171
Desenho 3 – 2ª camada condutora: PLANE1 172
Desenho 4 – 3ª camada condutora: MID1 173
Desenho 5 – 4ª camada condutora: PLANE2 174
Desenho 6 – 5ª camada condutora: MID2 175
Desenho 7 – 6ª camada condutora: PLANE3 176
Desenho 8 – 7ª camada condutora: PLANE4 177
Desenho 9 – 8ª camada condutora: MID3 178
Desenho 10 – 9ª camada condutora: PLANE5 179
Desenho 11 – 10ª camada condutora: MID4 180
Desenho 12 – 11ª camada condutora: PLANE6 181
Desenho 13 – 12ª camada condutora: BOTTOM 182
Desenho 14 – Camada de serigrafia superior: TOP SILK 183
Desenho 15 – Camada de serigrafia inferior: BOTTOM SILK 184



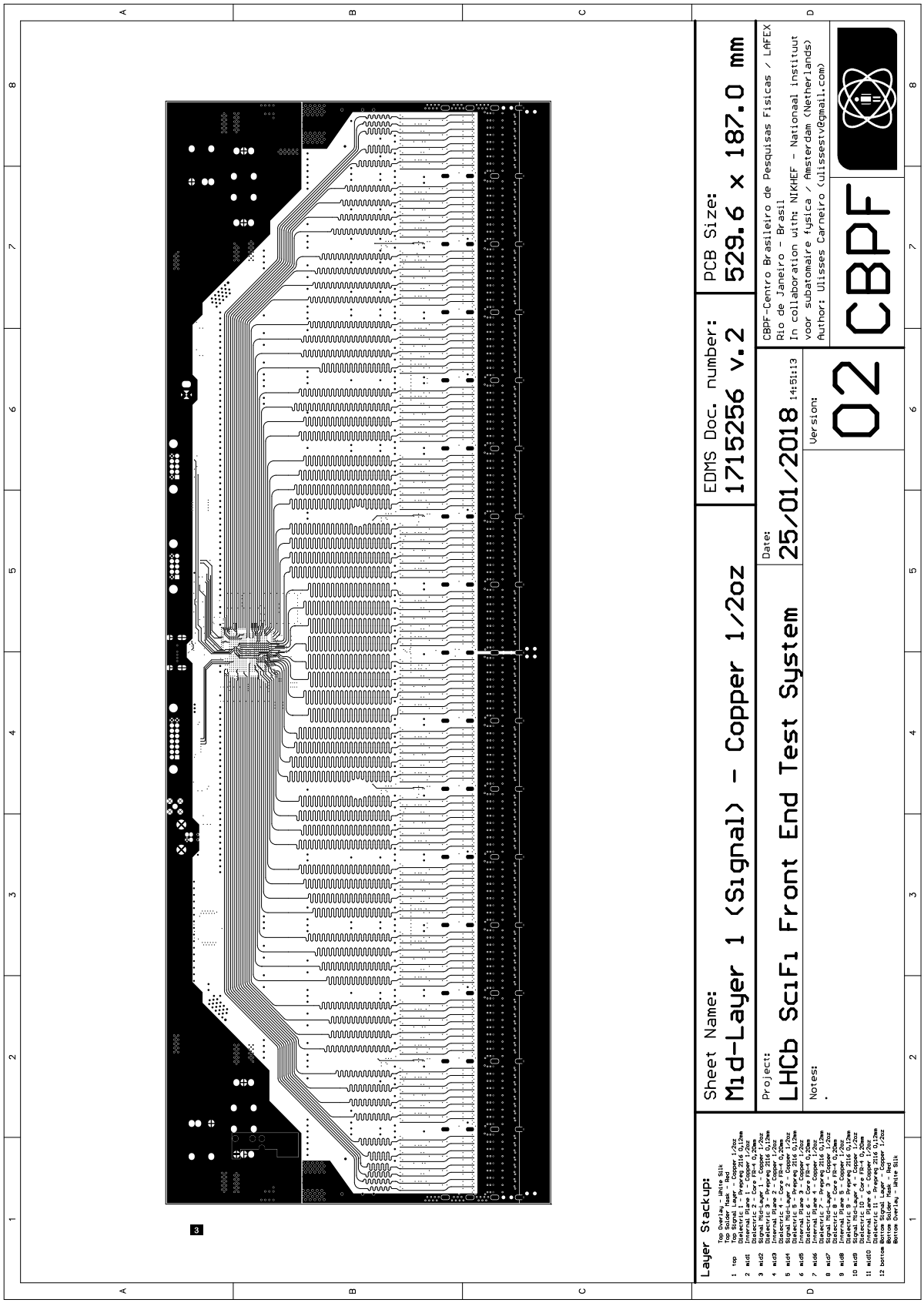
Desenho 1 – Dimensoões da Placa de Circuito Impresso



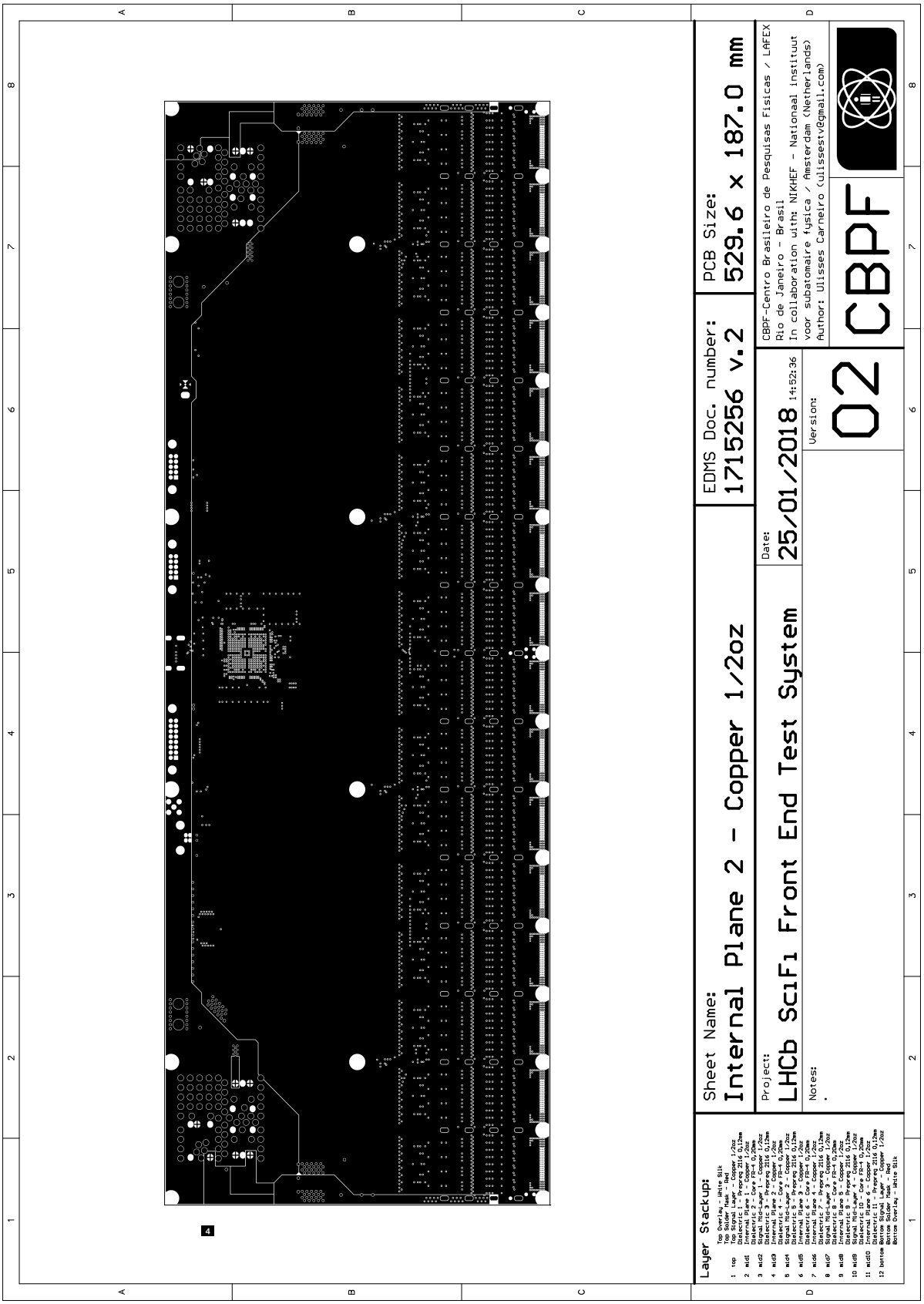
Desenho 2 – 1ª camada condutora: TOP



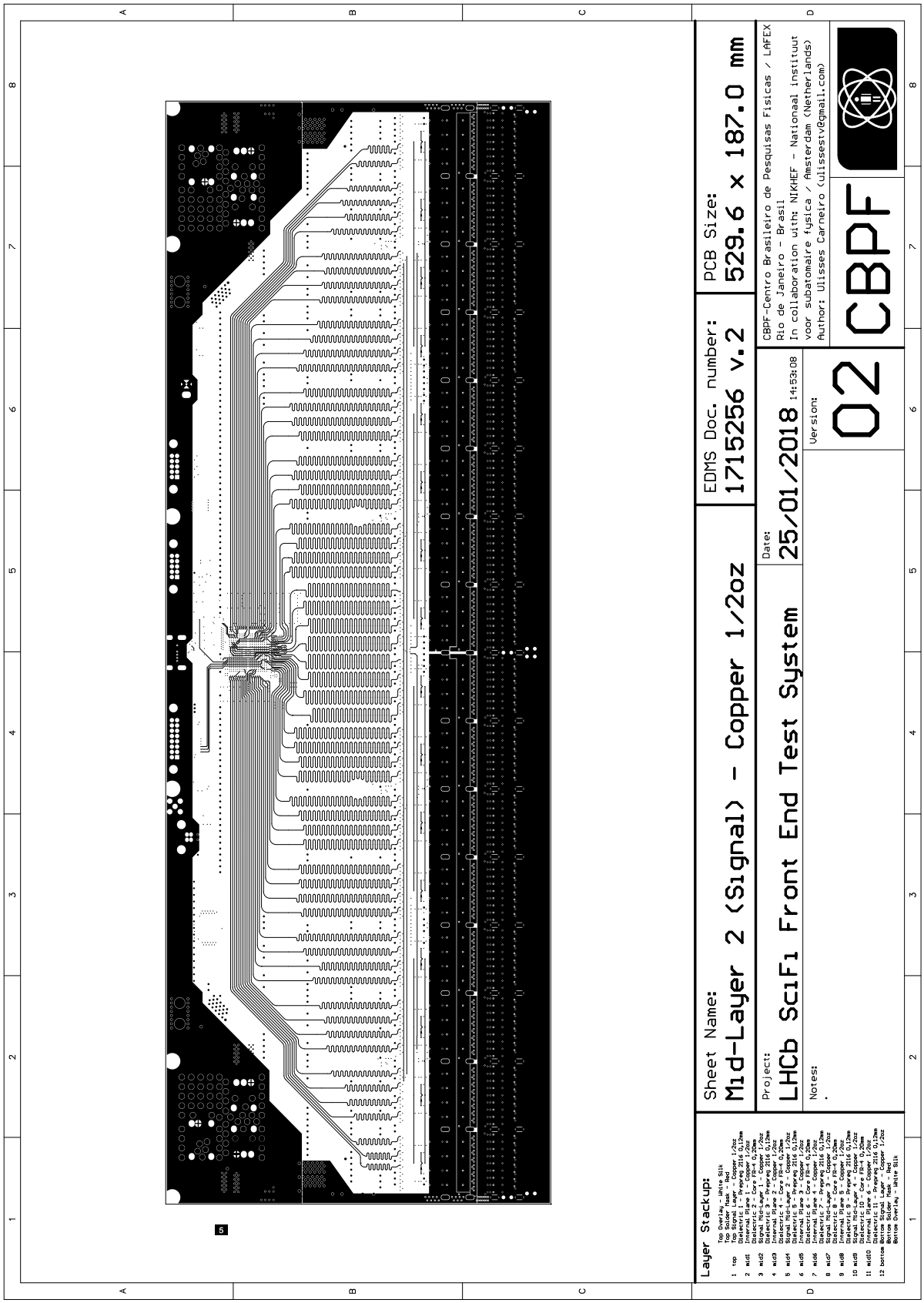
Desenho 3 – 2ª camada condutora: PLANE1



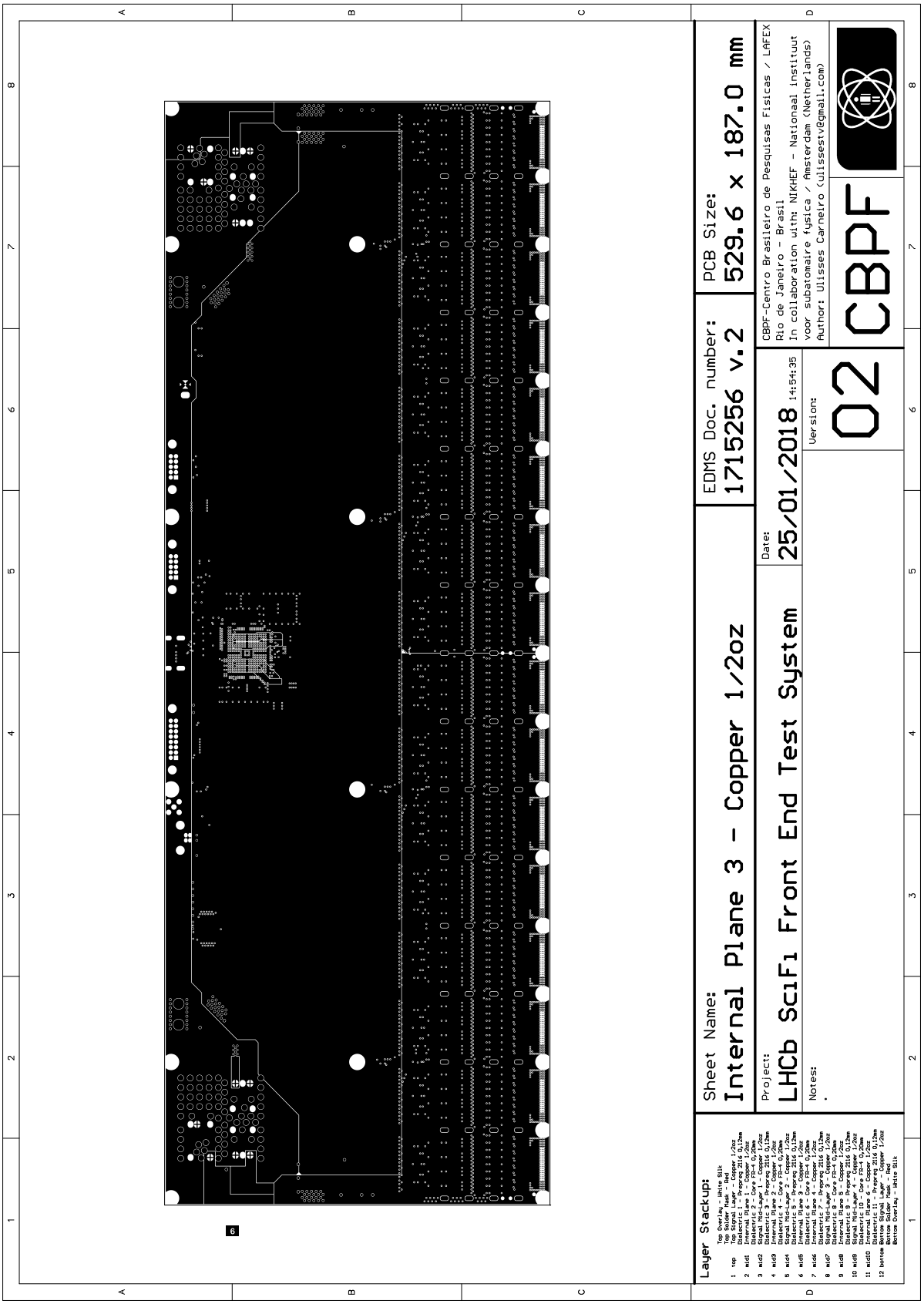
Desenho 4 – 3ª camada condutora: MID1



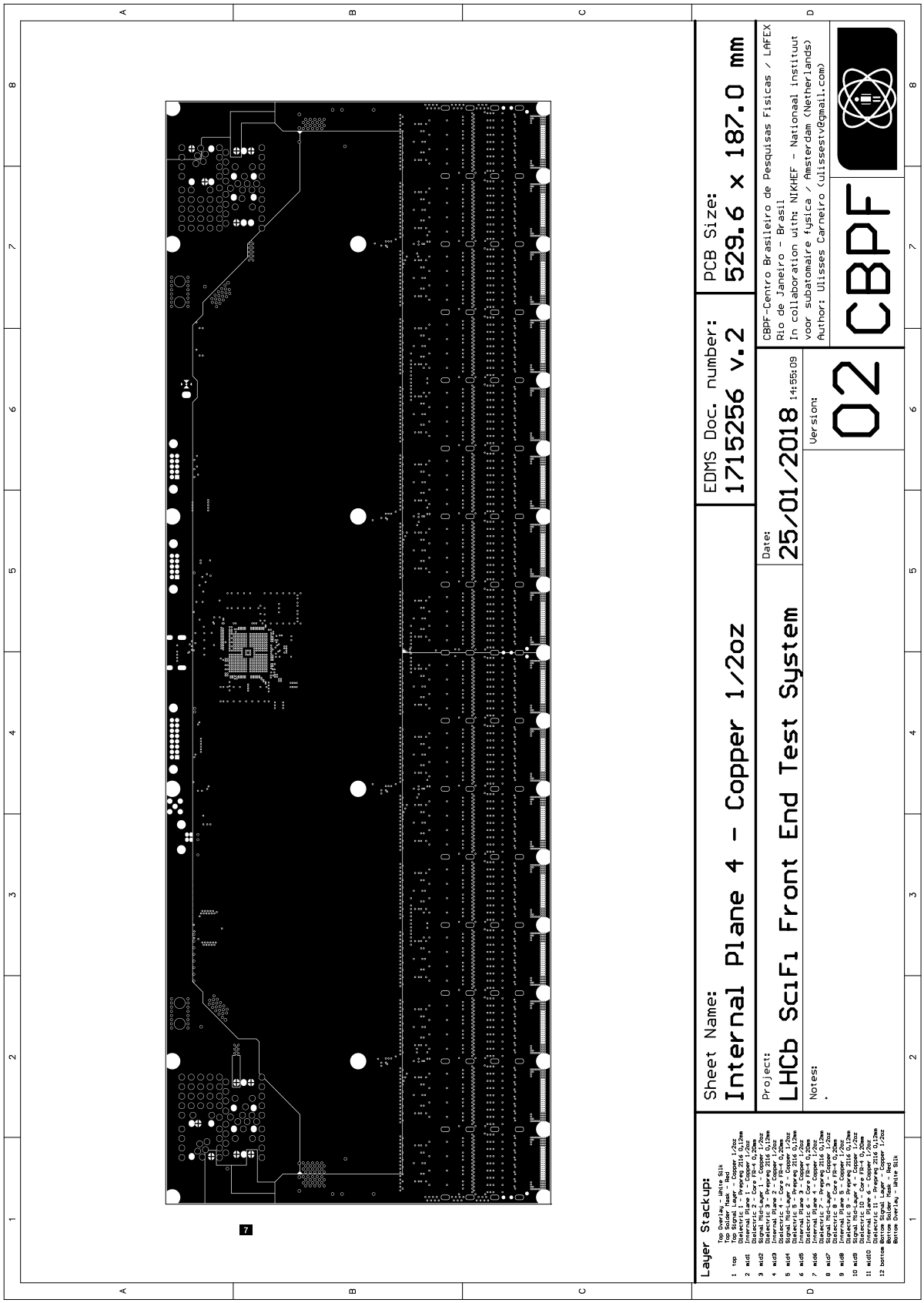
Desenho 5 – 4ª camada condutora: PLANE2



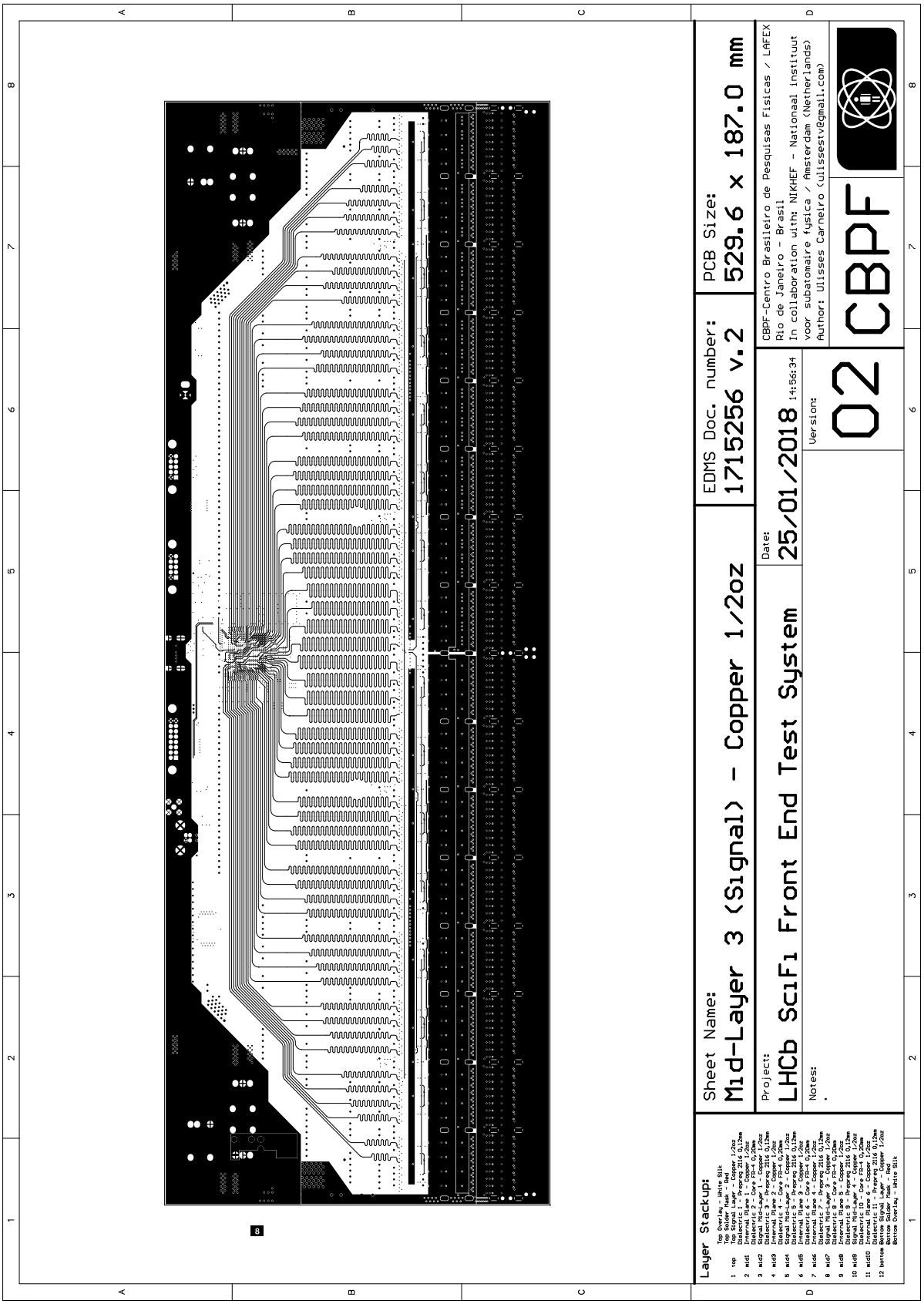
Desenho 6 – 5ª camada condutora: MID2



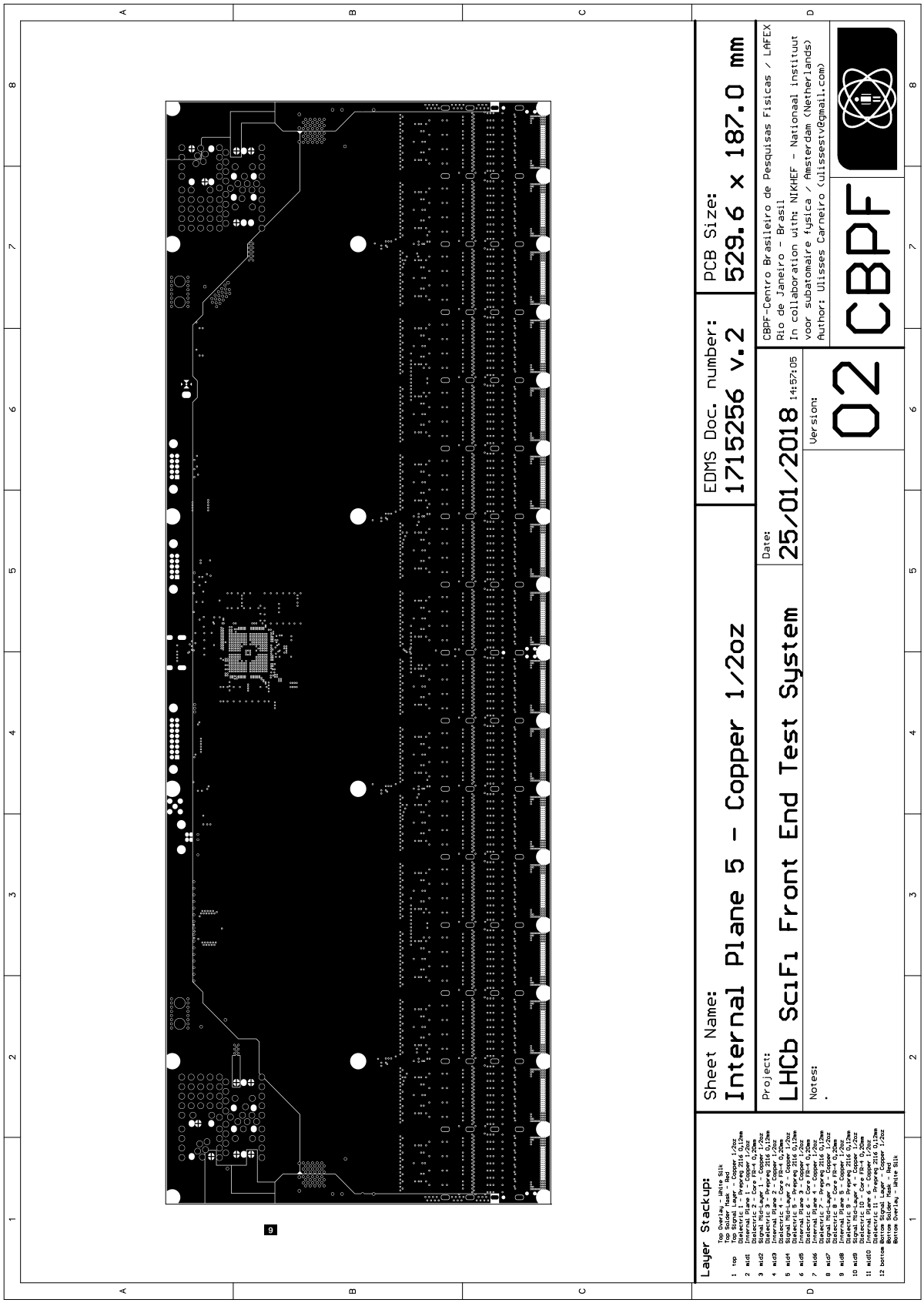
Desenho 7 – 6ª camada condutora: PLANE3



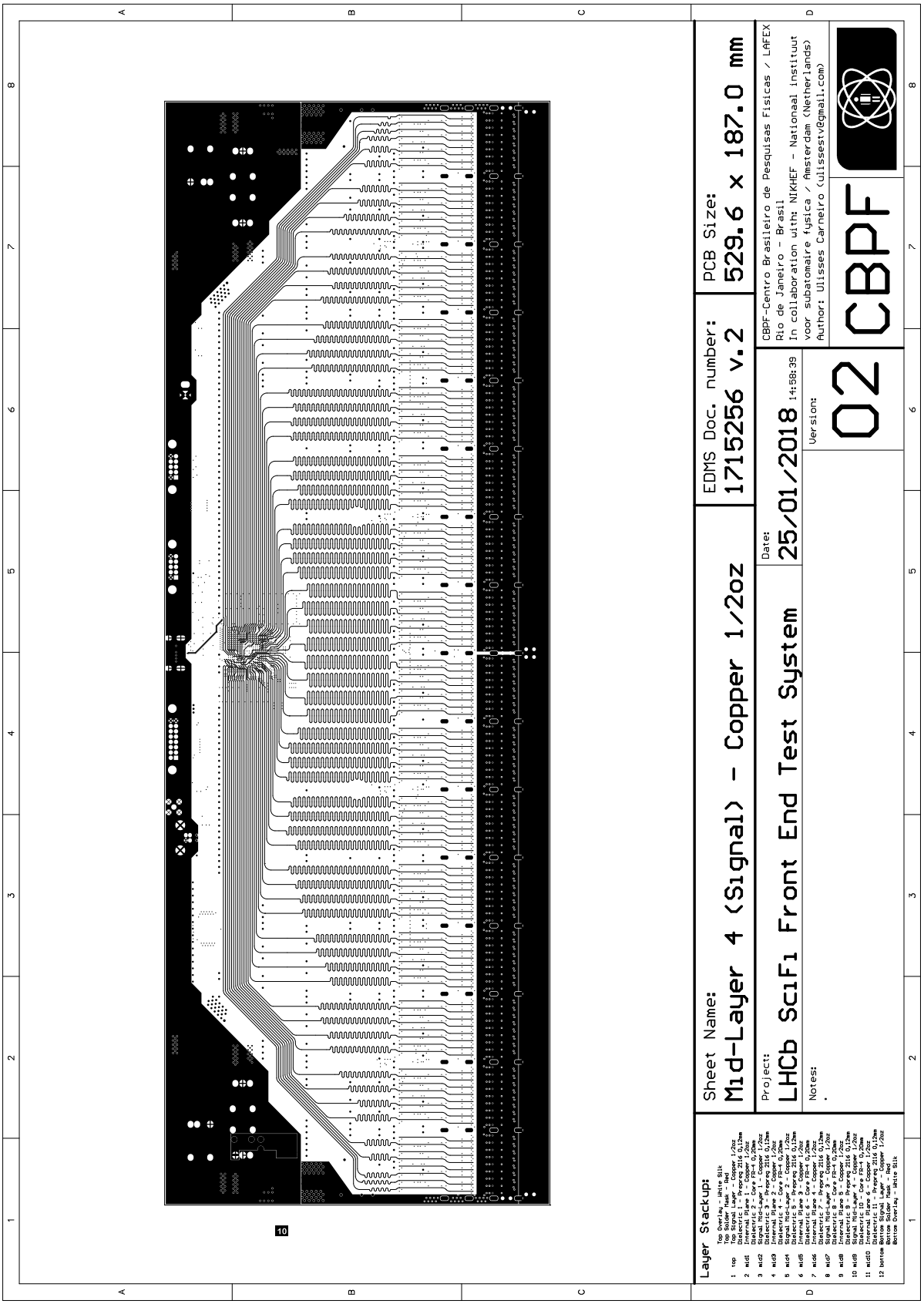
Desenho 8 – 7ª camada condutora: PLANE4



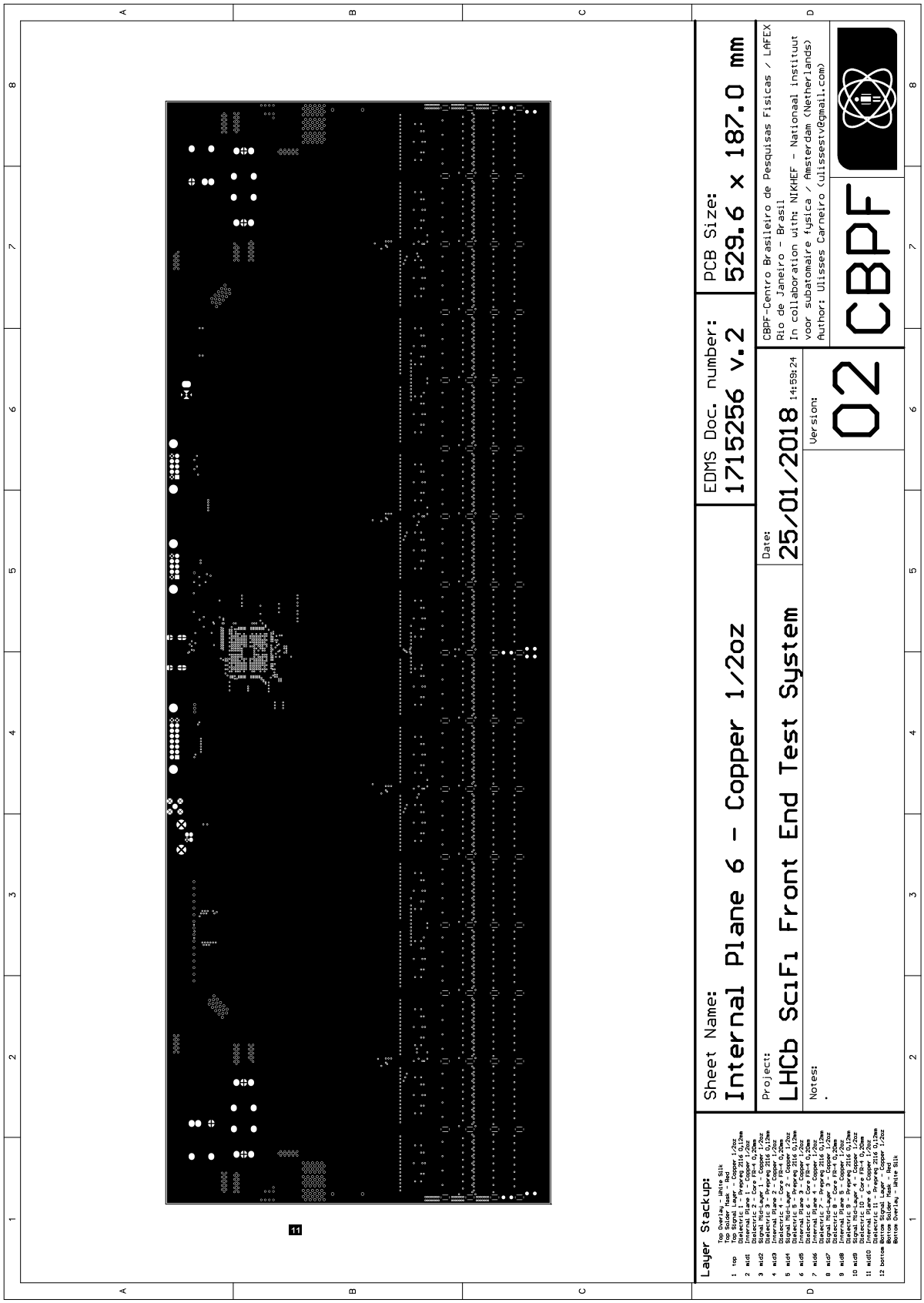
Desenho 9 – 8ª camada condutora: MID3



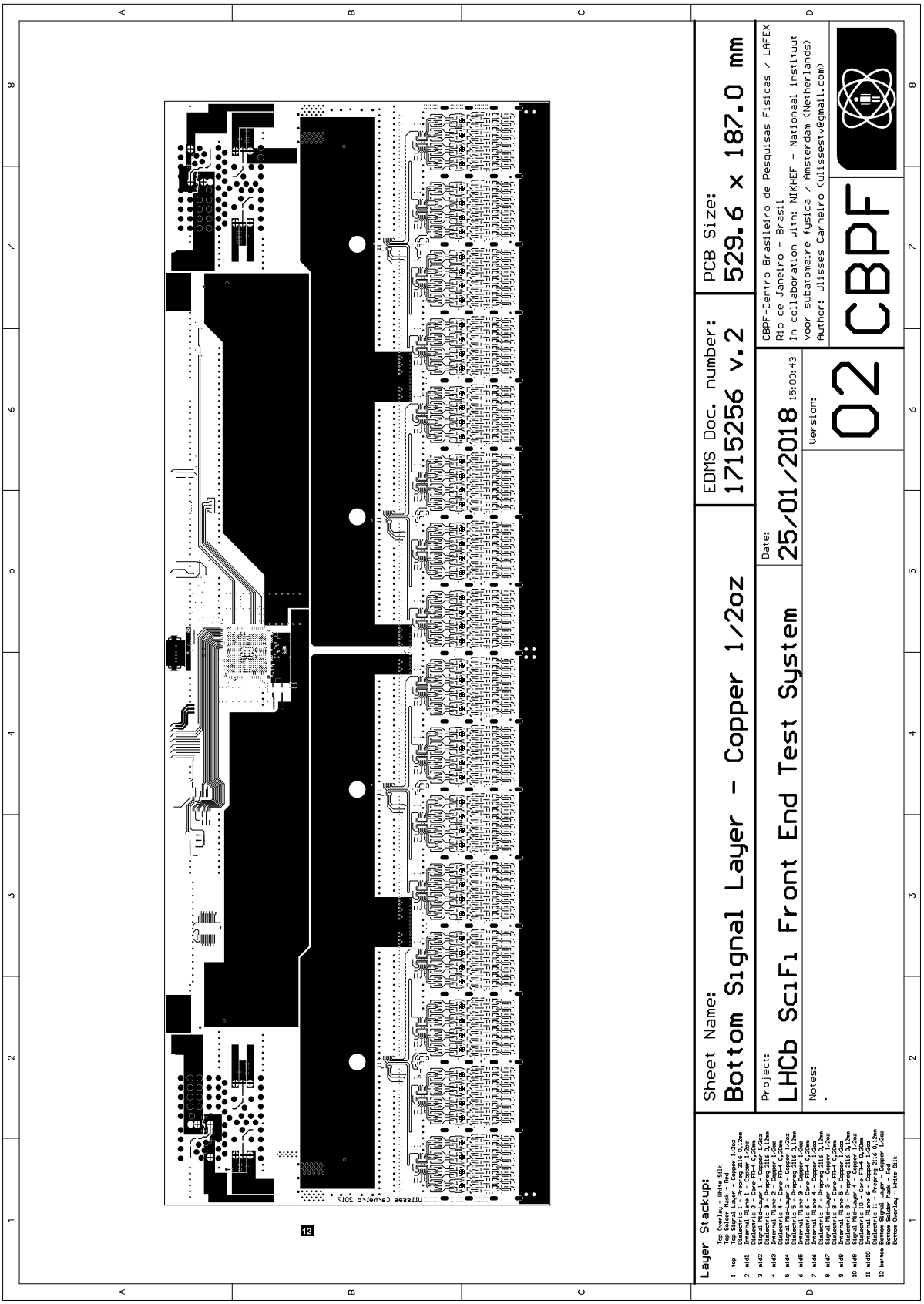
Desenho 10 – 9ª camada condutora: PLANE5



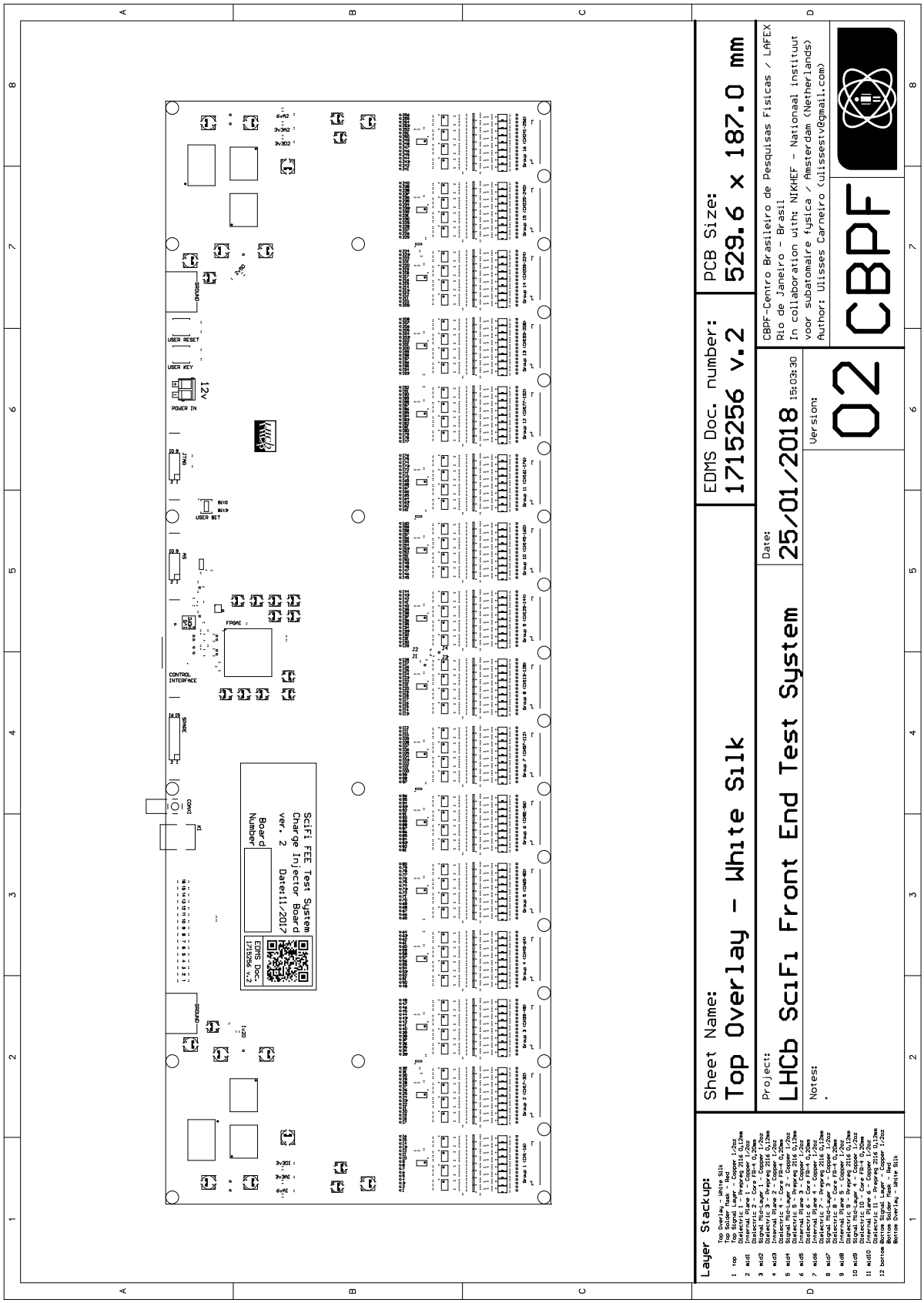
Desenho 11 – 10ª camada condutora: MID4



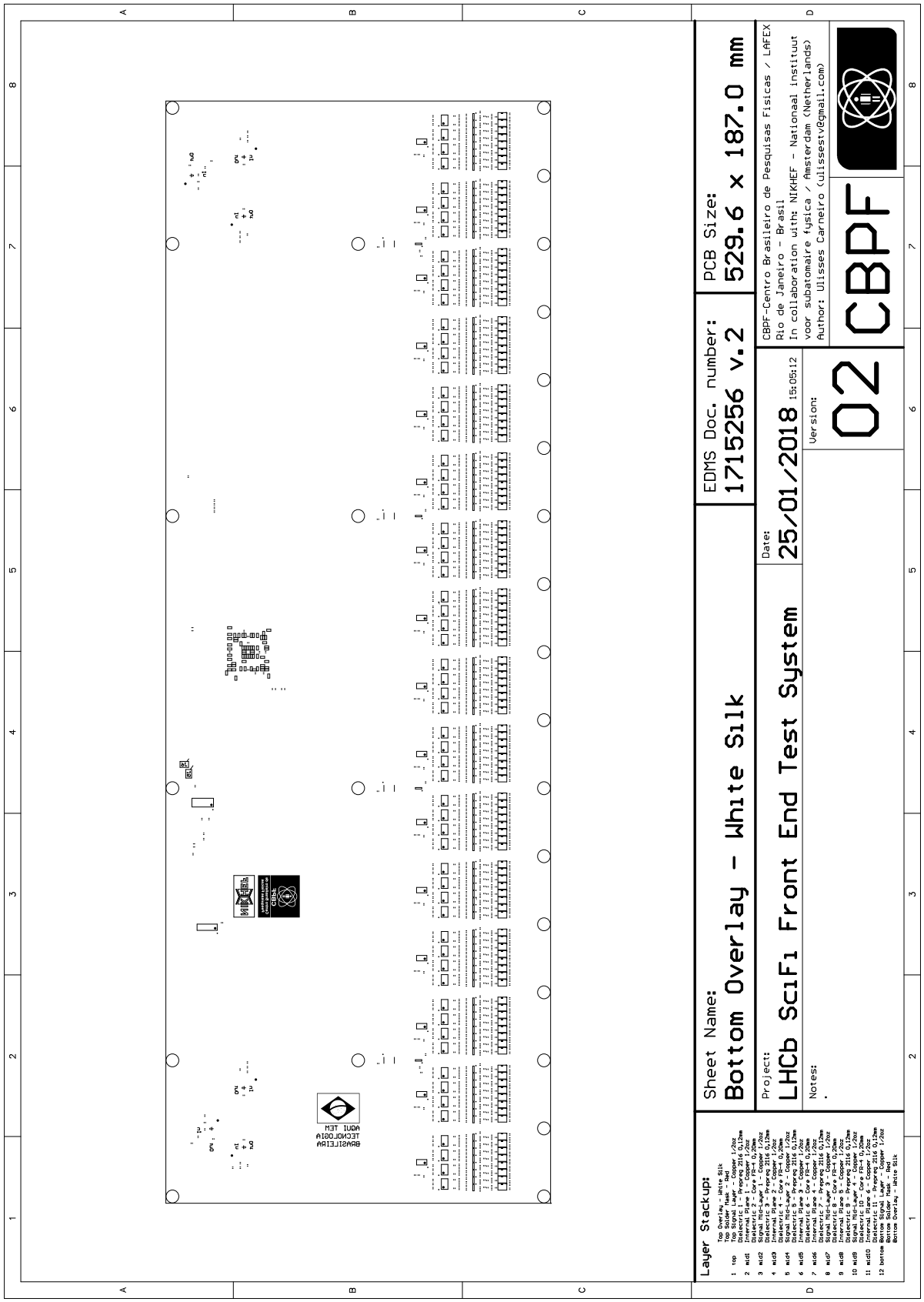
Desenho 12 – 11ª camada condutora: PLANE6



Desenho 13 – 12ª camada condutora: BOTTOM



Desenho 14 – Camada de serigrafia superior: TOP SILK



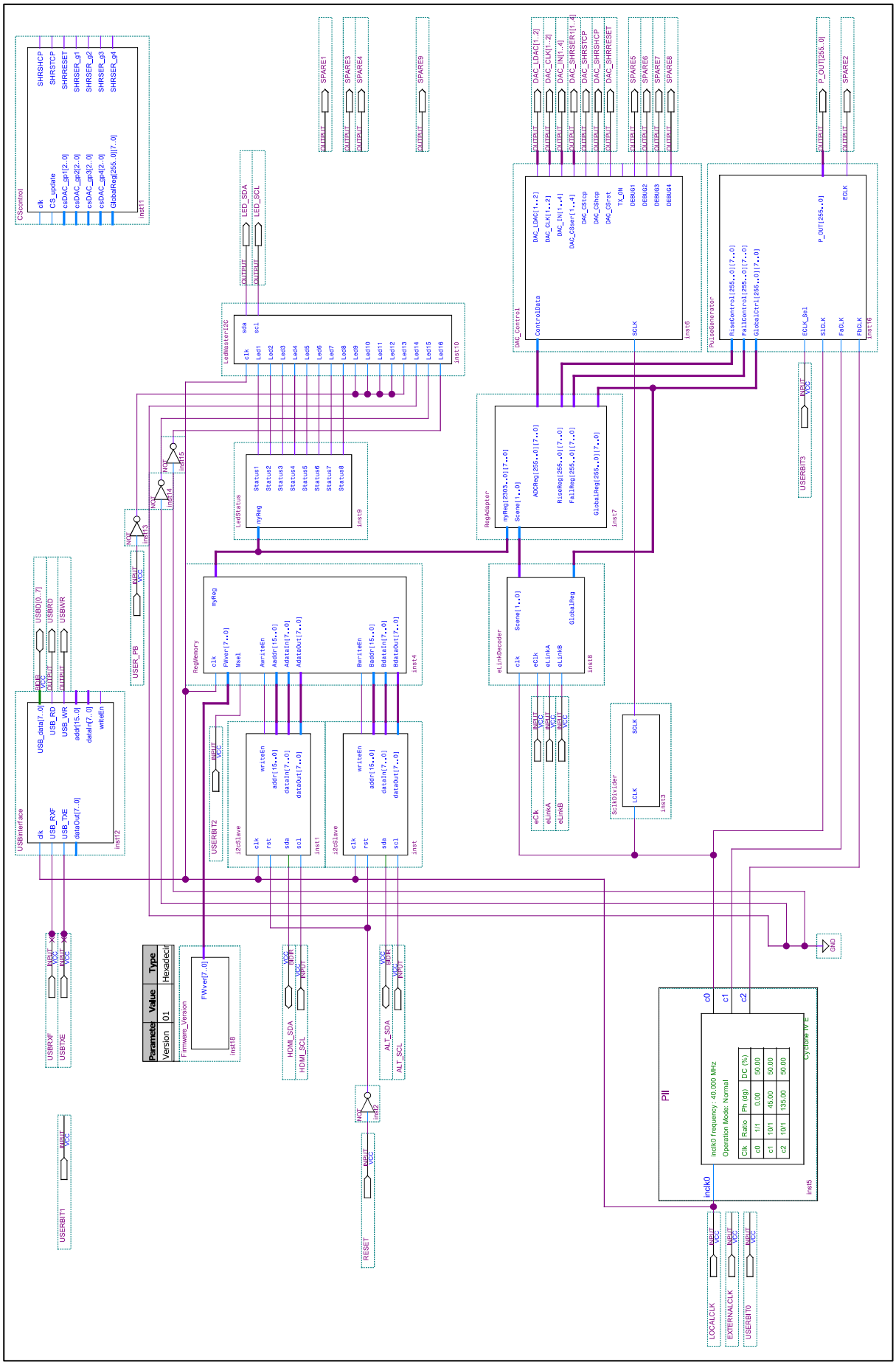
Desenho 15 – Camada de serigrafia inferior: BOTTOM SILK

Apêndice D

FIRMWARE do FPGA do Injetor de Carga

Arquivo 1 – InjProt.bdf (Top Level Entity) 186
Arquivo 2 - PulseGenerator.vhdl 187
Arquivo 3 - DAC_Control.vhdl 194
Arquivo 4 - RegMemory.vhdl 203
Arquivo 5 - RegAdapter.vhdl 205
Arquivo 6 - i2cSlave.v 206
Arquivo 7 - SclkDivider.vhdl 209
Arquivo 8 - eLinkDecoder.vhdl 210
Arquivo 9 - LedStatus.vhdl 212
Arquivo 10 - LedMasterI2C.vhdl 213
Arquivo 11 - UBSinterface.vhdl 218

Arquivo 1 – InjProt.bdf (Top Level Entity)



Arquivo 2 - PulseGenerator.vhdl

```
-- ******************************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:             LHCb SciFi Upgrade
-- Model:               Signal Injector Prototype
-- FPGA Proj. Name:     INJ_PROT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Ulisses Carneiro (ulissestv@gmail.com)
-- Date:                26-07-2015
-- ------------------------------------------------------------------------------
----
-- Module:              PulseGenerator
-- Description:         Pulse Generator for the Injector Board
--
********************************************************************************

-- ##############################################################################
-- Revision History:
--    Date     Author            Rev.    Comments
-- 26 Jul 15  Ulisses Carneiro  0001    Project and file creation
-- ##############################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

ENTITY PulseGenerator IS
 PORT(
  --Pulses Input/Output:
  SlCLK       : IN     std_logic;          --Slow Clock (40MHz  @   0 deg)
  FaCLK       : IN     std_logic;          --Fast Clock (400MHz @  45 deg)
  FbCLK       : IN     std_logic;          --Fast Clock (400MHz @ 135 deg)
  ECLK_Sel    : IN     std_logic;          --Trigger Out Source Selector
  ECLK        : OUT    std_logic;          --Trigger Out (NIM Output)
  P_OUT       : OUT    std_logic_vector (255 DOWNTO 0);--Pulse Output

 --Control:
  RiseControl : IN     SubArray;           --Channel 1 pulse control
  FallControl : IN     SubArray;           --Channel 1 pulse control
  GlobalCtrl  : IN     SubArray            --Channel 1 pulse control
 );

END PulseGenerator;

architecture circuit of PulseGenerator is

signal PULSE_GEN:std_logic;
signal PULSE_GEN1:std_logic;

signal FLAG_RST1:std_logic:='0';
signal FLAG_RST2:std_logic:='0';

signal ENABLE:std_logic;
signal RESET:std_logic;

signal count:unsigned (7 downto 0);
```

```
signal count0:unsigned (11 downto 0);

signal RiseCoarse:unsigned (11 downto 0);
signal FallCoarse:unsigned (11 downto 0);
signal RiseCoarse1:unsigned (11 downto 0);
signal FallCoarse1:unsigned (11 downto 0);

signal RepeatRate:unsigned (7 downto 0);

begin

ECLK    <= PULSE_GEN1;

RiseCoarse  <= "0000"&RiseControl(0)(7 downto 0);
RiseCoarse1 <= "0000"&RiseControl(1)(7 downto 0);

FallCoarse  <= "0000"&FallControl(0)(7 downto 0);
FallCoarse1 <= "0000"&FallControl(1)(7 downto 0);

RepeatRate <= GlobalCtrl(129); -- Register 0x0881

EnableLogic1:process(SlCLK)
begin
if SlCLK'event and SlCLK='0' then
  if count < RepeatRate then
    count <= count+1;
  else
    count <= conv_unsigned(0, 8);
  end if;
end if;
end process;

ResetLogic1:process(SlCLK,count)
begin
if SlCLK'event and SlCLK='1' and count=0 then
  FLAG_RST1 <= not FLAG_RST1;
end if;
end process;

ResetLogic2:process(FbCLK,RESET)
begin
if FbCLK'event and FbCLK='0' and RESET='1' then
  FLAG_RST2 <= not FLAG_RST2;
end if;
end process;

RESET <= FLAG_RST1 xor FLAG_RST2;

MainCounter:process(FaCLK)
begin
if FaCLK'event and FaCLK='0' then
  if (RESET = '1') then
    count0 <= conv_unsigned(0,12);
  else
    count0 <= count0 + conv_unsigned(1,12);
  end if;
end if;
end process;

PulseModulator:process(count0,FaCLK,FbCLK)
begin
if RiseCoarse >= FallCoarse then
```

```
  PULSE_GEN <= '0';
elsif FaCLK'event and FaCLK='0' then
  if count0 = FallCoarse then
    PULSE_GEN <= '0';
  elsif count0 = RiseCoarse then
    PULSE_GEN <= '1';
  end if;
end if;
end process;

PulseModulator1:process(count0,FaCLK,FbCLK)
begin
if RiseCoarse1 >= FallCoarse1 then
  PULSE_GEN1 <= '0';
elsif FaCLK'event and FaCLK='0' then
  if count0 = FallCoarse1 then
    PULSE_GEN1 <= '0';
  elsif count0 = RiseCoarse1 then
    PULSE_GEN1 <= '1';
  end if;
end if;
end process;

P_OUT(0)  <= (GlobalCtrl( 0)(0) and PULSE_GEN) xor GlobalCtrl(32)(0);
P_OUT(1)  <= (GlobalCtrl( 0)(1) and PULSE_GEN) xor GlobalCtrl(32)(1);
P_OUT(2)  <= (GlobalCtrl( 0)(2) and PULSE_GEN) xor GlobalCtrl(32)(2);
P_OUT(3)  <= (GlobalCtrl( 0)(3) and PULSE_GEN) xor GlobalCtrl(32)(3);
P_OUT(4)  <= (GlobalCtrl( 0)(4) and PULSE_GEN) xor GlobalCtrl(32)(4);
P_OUT(5)  <= (GlobalCtrl( 0)(5) and PULSE_GEN) xor GlobalCtrl(32)(5);
P_OUT(6)  <= (GlobalCtrl( 0)(6) and PULSE_GEN) xor GlobalCtrl(32)(6);
P_OUT(7)  <= (GlobalCtrl( 0)(7) and PULSE_GEN) xor GlobalCtrl(32)(7);
P_OUT(8)  <= (GlobalCtrl( 1)(0) and PULSE_GEN) xor GlobalCtrl(33)(0);
P_OUT(9)  <= (GlobalCtrl( 1)(1) and PULSE_GEN) xor GlobalCtrl(33)(1);
P_OUT(10) <= (GlobalCtrl( 1)(2) and PULSE_GEN) xor GlobalCtrl(33)(2);
P_OUT(11) <= (GlobalCtrl( 1)(3) and PULSE_GEN) xor GlobalCtrl(33)(3);
P_OUT(12) <= (GlobalCtrl( 1)(4) and PULSE_GEN) xor GlobalCtrl(33)(4);
P_OUT(13) <= (GlobalCtrl( 1)(5) and PULSE_GEN) xor GlobalCtrl(33)(5);
P_OUT(14) <= (GlobalCtrl( 1)(6) and PULSE_GEN) xor GlobalCtrl(33)(6);
P_OUT(15) <= (GlobalCtrl( 1)(7) and PULSE_GEN) xor GlobalCtrl(33)(7);
P_OUT(16) <= (GlobalCtrl( 2)(0) and PULSE_GEN) xor GlobalCtrl(34)(0);
P_OUT(17) <= (GlobalCtrl( 2)(1) and PULSE_GEN) xor GlobalCtrl(34)(1);
P_OUT(18) <= (GlobalCtrl( 2)(2) and PULSE_GEN) xor GlobalCtrl(34)(2);
P_OUT(19) <= (GlobalCtrl( 2)(3) and PULSE_GEN) xor GlobalCtrl(34)(3);
P_OUT(20) <= (GlobalCtrl( 2)(4) and PULSE_GEN) xor GlobalCtrl(34)(4);
P_OUT(21) <= (GlobalCtrl( 2)(5) and PULSE_GEN) xor GlobalCtrl(34)(5);
P_OUT(22) <= (GlobalCtrl( 2)(6) and PULSE_GEN) xor GlobalCtrl(34)(6);
P_OUT(23) <= (GlobalCtrl( 2)(7) and PULSE_GEN) xor GlobalCtrl(34)(7);
P_OUT(24) <= (GlobalCtrl( 3)(0) and PULSE_GEN) xor GlobalCtrl(35)(0);
P_OUT(25) <= (GlobalCtrl( 3)(1) and PULSE_GEN) xor GlobalCtrl(35)(1);
P_OUT(26) <= (GlobalCtrl( 3)(2) and PULSE_GEN) xor GlobalCtrl(35)(2);
P_OUT(27) <= (GlobalCtrl( 3)(3) and PULSE_GEN) xor GlobalCtrl(35)(3);
P_OUT(28) <= (GlobalCtrl( 3)(4) and PULSE_GEN) xor GlobalCtrl(35)(4);
P_OUT(29) <= (GlobalCtrl( 3)(5) and PULSE_GEN) xor GlobalCtrl(35)(5);
P_OUT(30) <= (GlobalCtrl( 3)(6) and PULSE_GEN) xor GlobalCtrl(35)(6);
P_OUT(31) <= (GlobalCtrl( 3)(7) and PULSE_GEN) xor GlobalCtrl(35)(7);
P_OUT(32) <= (GlobalCtrl( 4)(0) and PULSE_GEN) xor GlobalCtrl(36)(0);
P_OUT(33) <= (GlobalCtrl( 4)(1) and PULSE_GEN) xor GlobalCtrl(36)(1);
P_OUT(34) <= (GlobalCtrl( 4)(2) and PULSE_GEN) xor GlobalCtrl(36)(2);
P_OUT(35) <= (GlobalCtrl( 4)(3) and PULSE_GEN) xor GlobalCtrl(36)(3);
P_OUT(36) <= (GlobalCtrl( 4)(4) and PULSE_GEN) xor GlobalCtrl(36)(4);
P_OUT(37) <= (GlobalCtrl( 4)(5) and PULSE_GEN) xor GlobalCtrl(36)(5);
P_OUT(38) <= (GlobalCtrl( 4)(6) and PULSE_GEN) xor GlobalCtrl(36)(6);
P_OUT(39) <= (GlobalCtrl( 4)(7) and PULSE_GEN) xor GlobalCtrl(36)(7);
```

```
P_OUT(40)  <= (GlobalCtrl( 5)(0) and PULSE_GEN) xor GlobalCtrl(37)(0);
P_OUT(41)  <= (GlobalCtrl( 5)(1) and PULSE_GEN) xor GlobalCtrl(37)(1);
P_OUT(42)  <= (GlobalCtrl( 5)(2) and PULSE_GEN) xor GlobalCtrl(37)(2);
P_OUT(43)  <= (GlobalCtrl( 5)(3) and PULSE_GEN) xor GlobalCtrl(37)(3);
P_OUT(44)  <= (GlobalCtrl( 5)(4) and PULSE_GEN) xor GlobalCtrl(37)(4);
P_OUT(45)  <= (GlobalCtrl( 5)(5) and PULSE_GEN) xor GlobalCtrl(37)(5);
P_OUT(46)  <= (GlobalCtrl( 5)(6) and PULSE_GEN) xor GlobalCtrl(37)(6);
P_OUT(47)  <= (GlobalCtrl( 5)(7) and PULSE_GEN) xor GlobalCtrl(37)(7);
P_OUT(48)  <= (GlobalCtrl( 6)(0) and PULSE_GEN) xor GlobalCtrl(38)(0);
P_OUT(49)  <= (GlobalCtrl( 6)(1) and PULSE_GEN) xor GlobalCtrl(38)(1);
P_OUT(50)  <= (GlobalCtrl( 6)(2) and PULSE_GEN) xor GlobalCtrl(38)(2);
P_OUT(51)  <= (GlobalCtrl( 6)(3) and PULSE_GEN) xor GlobalCtrl(38)(3);
P_OUT(52)  <= (GlobalCtrl( 6)(4) and PULSE_GEN) xor GlobalCtrl(38)(4);
P_OUT(53)  <= (GlobalCtrl( 6)(5) and PULSE_GEN) xor GlobalCtrl(38)(5);
P_OUT(54)  <= (GlobalCtrl( 6)(6) and PULSE_GEN) xor GlobalCtrl(38)(6);
P_OUT(55)  <= (GlobalCtrl( 6)(7) and PULSE_GEN) xor GlobalCtrl(38)(7);
P_OUT(56)  <= (GlobalCtrl( 7)(0) and PULSE_GEN) xor GlobalCtrl(39)(0);
P_OUT(57)  <= (GlobalCtrl( 7)(1) and PULSE_GEN) xor GlobalCtrl(39)(1);
P_OUT(58)  <= (GlobalCtrl( 7)(2) and PULSE_GEN) xor GlobalCtrl(39)(2);
P_OUT(59)  <= (GlobalCtrl( 7)(3) and PULSE_GEN) xor GlobalCtrl(39)(3);
P_OUT(60)  <= (GlobalCtrl( 7)(4) and PULSE_GEN) xor GlobalCtrl(39)(4);
P_OUT(61)  <= (GlobalCtrl( 7)(5) and PULSE_GEN) xor GlobalCtrl(39)(5);
P_OUT(62)  <= (GlobalCtrl( 7)(6) and PULSE_GEN) xor GlobalCtrl(39)(6);
P_OUT(63)  <= (GlobalCtrl( 7)(7) and PULSE_GEN) xor GlobalCtrl(39)(7);
P_OUT(64)  <= (GlobalCtrl( 8)(0) and PULSE_GEN) xor GlobalCtrl(40)(0);
P_OUT(65)  <= (GlobalCtrl( 8)(1) and PULSE_GEN) xor GlobalCtrl(40)(1);
P_OUT(66)  <= (GlobalCtrl( 8)(2) and PULSE_GEN) xor GlobalCtrl(40)(2);
P_OUT(67)  <= (GlobalCtrl( 8)(3) and PULSE_GEN) xor GlobalCtrl(40)(3);
P_OUT(68)  <= (GlobalCtrl( 8)(4) and PULSE_GEN) xor GlobalCtrl(40)(4);
P_OUT(69)  <= (GlobalCtrl( 8)(5) and PULSE_GEN) xor GlobalCtrl(40)(5);
P_OUT(70)  <= (GlobalCtrl( 8)(6) and PULSE_GEN) xor GlobalCtrl(40)(6);
P_OUT(71)  <= (GlobalCtrl( 8)(7) and PULSE_GEN) xor GlobalCtrl(40)(7);
P_OUT(72)  <= (GlobalCtrl( 9)(0) and PULSE_GEN) xor GlobalCtrl(41)(0);
P_OUT(73)  <= (GlobalCtrl( 9)(1) and PULSE_GEN) xor GlobalCtrl(41)(1);
P_OUT(74)  <= (GlobalCtrl( 9)(2) and PULSE_GEN) xor GlobalCtrl(41)(2);
P_OUT(75)  <= (GlobalCtrl( 9)(3) and PULSE_GEN) xor GlobalCtrl(41)(3);
P_OUT(76)  <= (GlobalCtrl( 9)(4) and PULSE_GEN) xor GlobalCtrl(41)(4);
P_OUT(77)  <= (GlobalCtrl( 9)(5) and PULSE_GEN) xor GlobalCtrl(41)(5);
P_OUT(78)  <= (GlobalCtrl( 9)(6) and PULSE_GEN) xor GlobalCtrl(41)(6);
P_OUT(79)  <= (GlobalCtrl( 9)(7) and PULSE_GEN) xor GlobalCtrl(41)(7);
P_OUT(80)  <= (GlobalCtrl(10)(0) and PULSE_GEN) xor GlobalCtrl(42)(0);
P_OUT(81)  <= (GlobalCtrl(10)(1) and PULSE_GEN) xor GlobalCtrl(42)(1);
P_OUT(82)  <= (GlobalCtrl(10)(2) and PULSE_GEN) xor GlobalCtrl(42)(2);
P_OUT(83)  <= (GlobalCtrl(10)(3) and PULSE_GEN) xor GlobalCtrl(42)(3);
P_OUT(84)  <= (GlobalCtrl(10)(4) and PULSE_GEN) xor GlobalCtrl(42)(4);
P_OUT(85)  <= (GlobalCtrl(10)(5) and PULSE_GEN) xor GlobalCtrl(42)(5);
P_OUT(86)  <= (GlobalCtrl(10)(6) and PULSE_GEN) xor GlobalCtrl(42)(6);
P_OUT(87)  <= (GlobalCtrl(10)(7) and PULSE_GEN) xor GlobalCtrl(42)(7);
P_OUT(88)  <= (GlobalCtrl(11)(0) and PULSE_GEN) xor GlobalCtrl(43)(0);
P_OUT(89)  <= (GlobalCtrl(11)(1) and PULSE_GEN) xor GlobalCtrl(43)(1);
P_OUT(90)  <= (GlobalCtrl(11)(2) and PULSE_GEN) xor GlobalCtrl(43)(2);
P_OUT(91)  <= (GlobalCtrl(11)(3) and PULSE_GEN) xor GlobalCtrl(43)(3);
P_OUT(92)  <= (GlobalCtrl(11)(4) and PULSE_GEN) xor GlobalCtrl(43)(4);
P_OUT(93)  <= (GlobalCtrl(11)(5) and PULSE_GEN) xor GlobalCtrl(43)(5);
P_OUT(94)  <= (GlobalCtrl(11)(6) and PULSE_GEN) xor GlobalCtrl(43)(6);
P_OUT(95)  <= (GlobalCtrl(11)(7) and PULSE_GEN) xor GlobalCtrl(43)(7);
P_OUT(96)  <= (GlobalCtrl(12)(0) and PULSE_GEN) xor GlobalCtrl(44)(0);
P_OUT(97)  <= (GlobalCtrl(12)(1) and PULSE_GEN) xor GlobalCtrl(44)(1);
P_OUT(98)  <= (GlobalCtrl(12)(2) and PULSE_GEN) xor GlobalCtrl(44)(2);
P_OUT(99)  <= (GlobalCtrl(12)(3) and PULSE_GEN) xor GlobalCtrl(44)(3);
P_OUT(100) <= (GlobalCtrl(12)(4) and PULSE_GEN) xor GlobalCtrl(44)(4);
P_OUT(101) <= (GlobalCtrl(12)(5) and PULSE_GEN) xor GlobalCtrl(44)(5);
P_OUT(102) <= (GlobalCtrl(12)(6) and PULSE_GEN) xor GlobalCtrl(44)(6);
```

```
P_OUT(103) <= (GlobalCtrl(12)(7) and PULSE_GEN) xor GlobalCtrl(44)(7);
P_OUT(104) <= (GlobalCtrl(13)(0) and PULSE_GEN) xor GlobalCtrl(45)(0);
P_OUT(105) <= (GlobalCtrl(13)(1) and PULSE_GEN) xor GlobalCtrl(45)(1);
P_OUT(106) <= (GlobalCtrl(13)(2) and PULSE_GEN) xor GlobalCtrl(45)(2);
P_OUT(107) <= (GlobalCtrl(13)(3) and PULSE_GEN) xor GlobalCtrl(45)(3);
P_OUT(108) <= (GlobalCtrl(13)(4) and PULSE_GEN) xor GlobalCtrl(45)(4);
P_OUT(109) <= (GlobalCtrl(13)(5) and PULSE_GEN) xor GlobalCtrl(45)(5);
P_OUT(110) <= (GlobalCtrl(13)(6) and PULSE_GEN) xor GlobalCtrl(45)(6);
P_OUT(111) <= (GlobalCtrl(13)(7) and PULSE_GEN) xor GlobalCtrl(45)(7);
P_OUT(112) <= (GlobalCtrl(14)(0) and PULSE_GEN) xor GlobalCtrl(46)(0);
P_OUT(113) <= (GlobalCtrl(14)(1) and PULSE_GEN) xor GlobalCtrl(46)(1);
P_OUT(114) <= (GlobalCtrl(14)(2) and PULSE_GEN) xor GlobalCtrl(46)(2);
P_OUT(115) <= (GlobalCtrl(14)(3) and PULSE_GEN) xor GlobalCtrl(46)(3);
P_OUT(116) <= (GlobalCtrl(14)(4) and PULSE_GEN) xor GlobalCtrl(46)(4);
P_OUT(117) <= (GlobalCtrl(14)(5) and PULSE_GEN) xor GlobalCtrl(46)(5);
P_OUT(118) <= (GlobalCtrl(14)(6) and PULSE_GEN) xor GlobalCtrl(46)(6);
P_OUT(119) <= (GlobalCtrl(14)(7) and PULSE_GEN) xor GlobalCtrl(46)(7);
P_OUT(120) <= (GlobalCtrl(15)(0) and PULSE_GEN) xor GlobalCtrl(47)(0);
P_OUT(121) <= (GlobalCtrl(15)(1) and PULSE_GEN) xor GlobalCtrl(47)(1);
P_OUT(122) <= (GlobalCtrl(15)(2) and PULSE_GEN) xor GlobalCtrl(47)(2);
P_OUT(123) <= (GlobalCtrl(15)(3) and PULSE_GEN) xor GlobalCtrl(47)(3);
P_OUT(124) <= (GlobalCtrl(15)(4) and PULSE_GEN) xor GlobalCtrl(47)(4);
P_OUT(125) <= (GlobalCtrl(15)(5) and PULSE_GEN) xor GlobalCtrl(47)(5);
P_OUT(126) <= (GlobalCtrl(15)(6) and PULSE_GEN) xor GlobalCtrl(47)(6);
P_OUT(127) <= (GlobalCtrl(15)(7) and PULSE_GEN) xor GlobalCtrl(47)(7);
P_OUT(128) <= (GlobalCtrl(16)(0) and PULSE_GEN) xor GlobalCtrl(48)(0);
P_OUT(129) <= (GlobalCtrl(16)(1) and PULSE_GEN) xor GlobalCtrl(48)(1);
P_OUT(130) <= (GlobalCtrl(16)(2) and PULSE_GEN) xor GlobalCtrl(48)(2);
P_OUT(131) <= (GlobalCtrl(16)(3) and PULSE_GEN) xor GlobalCtrl(48)(3);
P_OUT(132) <= (GlobalCtrl(16)(4) and PULSE_GEN) xor GlobalCtrl(48)(4);
P_OUT(133) <= (GlobalCtrl(16)(5) and PULSE_GEN) xor GlobalCtrl(48)(5);
P_OUT(134) <= (GlobalCtrl(16)(6) and PULSE_GEN) xor GlobalCtrl(48)(6);
P_OUT(135) <= (GlobalCtrl(16)(7) and PULSE_GEN) xor GlobalCtrl(48)(7);
P_OUT(136) <= (GlobalCtrl(17)(0) and PULSE_GEN) xor GlobalCtrl(49)(0);
P_OUT(137) <= (GlobalCtrl(17)(1) and PULSE_GEN) xor GlobalCtrl(49)(1);
P_OUT(138) <= (GlobalCtrl(17)(2) and PULSE_GEN) xor GlobalCtrl(49)(2);
P_OUT(139) <= (GlobalCtrl(17)(3) and PULSE_GEN) xor GlobalCtrl(49)(3);
P_OUT(140) <= (GlobalCtrl(17)(4) and PULSE_GEN) xor GlobalCtrl(49)(4);
P_OUT(141) <= (GlobalCtrl(17)(5) and PULSE_GEN) xor GlobalCtrl(49)(5);
P_OUT(142) <= (GlobalCtrl(17)(6) and PULSE_GEN) xor GlobalCtrl(49)(6);
P_OUT(143) <= (GlobalCtrl(17)(7) and PULSE_GEN) xor GlobalCtrl(49)(7);
P_OUT(144) <= (GlobalCtrl(18)(0) and PULSE_GEN) xor GlobalCtrl(50)(0);
P_OUT(145) <= (GlobalCtrl(18)(1) and PULSE_GEN) xor GlobalCtrl(50)(1);
P_OUT(146) <= (GlobalCtrl(18)(2) and PULSE_GEN) xor GlobalCtrl(50)(2);
P_OUT(147) <= (GlobalCtrl(18)(3) and PULSE_GEN) xor GlobalCtrl(50)(3);
P_OUT(148) <= (GlobalCtrl(18)(4) and PULSE_GEN) xor GlobalCtrl(50)(4);
P_OUT(149) <= (GlobalCtrl(18)(5) and PULSE_GEN) xor GlobalCtrl(50)(5);
P_OUT(150) <= (GlobalCtrl(18)(6) and PULSE_GEN) xor GlobalCtrl(50)(6);
P_OUT(151) <= (GlobalCtrl(18)(7) and PULSE_GEN) xor GlobalCtrl(50)(7);
P_OUT(152) <= (GlobalCtrl(19)(0) and PULSE_GEN) xor GlobalCtrl(51)(0);
P_OUT(153) <= (GlobalCtrl(19)(1) and PULSE_GEN) xor GlobalCtrl(51)(1);
P_OUT(154) <= (GlobalCtrl(19)(2) and PULSE_GEN) xor GlobalCtrl(51)(2);
P_OUT(155) <= (GlobalCtrl(19)(3) and PULSE_GEN) xor GlobalCtrl(51)(3);
P_OUT(156) <= (GlobalCtrl(19)(4) and PULSE_GEN) xor GlobalCtrl(51)(4);
P_OUT(157) <= (GlobalCtrl(19)(5) and PULSE_GEN) xor GlobalCtrl(51)(5);
P_OUT(158) <= (GlobalCtrl(19)(6) and PULSE_GEN) xor GlobalCtrl(51)(6);
P_OUT(159) <= (GlobalCtrl(19)(7) and PULSE_GEN) xor GlobalCtrl(51)(7);
P_OUT(160) <= (GlobalCtrl(20)(0) and PULSE_GEN) xor GlobalCtrl(52)(0);
P_OUT(161) <= (GlobalCtrl(20)(1) and PULSE_GEN) xor GlobalCtrl(52)(1);
P_OUT(162) <= (GlobalCtrl(20)(2) and PULSE_GEN) xor GlobalCtrl(52)(2);
P_OUT(163) <= (GlobalCtrl(20)(3) and PULSE_GEN) xor GlobalCtrl(52)(3);
P_OUT(164) <= (GlobalCtrl(20)(4) and PULSE_GEN) xor GlobalCtrl(52)(4);
P_OUT(165) <= (GlobalCtrl(20)(5) and PULSE_GEN) xor GlobalCtrl(52)(5);
```

```
P_OUT(166) <= (GlobalCtrl(20)(6) and PULSE_GEN) xor GlobalCtrl(52)(6);
P_OUT(167) <= (GlobalCtrl(20)(7) and PULSE_GEN) xor GlobalCtrl(52)(7);
P_OUT(168) <= (GlobalCtrl(21)(0) and PULSE_GEN) xor GlobalCtrl(53)(0);
P_OUT(169) <= (GlobalCtrl(21)(1) and PULSE_GEN) xor GlobalCtrl(53)(1);
P_OUT(170) <= (GlobalCtrl(21)(2) and PULSE_GEN) xor GlobalCtrl(53)(2);
P_OUT(171) <= (GlobalCtrl(21)(3) and PULSE_GEN) xor GlobalCtrl(53)(3);
P_OUT(172) <= (GlobalCtrl(21)(4) and PULSE_GEN) xor GlobalCtrl(53)(4);
P_OUT(173) <= (GlobalCtrl(21)(5) and PULSE_GEN) xor GlobalCtrl(53)(5);
P_OUT(174) <= (GlobalCtrl(21)(6) and PULSE_GEN) xor GlobalCtrl(53)(6);
P_OUT(175) <= (GlobalCtrl(21)(7) and PULSE_GEN) xor GlobalCtrl(53)(7);
P_OUT(176) <= (GlobalCtrl(22)(0) and PULSE_GEN) xor GlobalCtrl(54)(0);
P_OUT(177) <= (GlobalCtrl(22)(1) and PULSE_GEN) xor GlobalCtrl(54)(1);
P_OUT(178) <= (GlobalCtrl(22)(2) and PULSE_GEN) xor GlobalCtrl(54)(2);
P_OUT(179) <= (GlobalCtrl(22)(3) and PULSE_GEN) xor GlobalCtrl(54)(3);
P_OUT(180) <= (GlobalCtrl(22)(4) and PULSE_GEN) xor GlobalCtrl(54)(4);
P_OUT(181) <= (GlobalCtrl(22)(5) and PULSE_GEN) xor GlobalCtrl(54)(5);
P_OUT(182) <= (GlobalCtrl(22)(6) and PULSE_GEN) xor GlobalCtrl(54)(6);
P_OUT(183) <= (GlobalCtrl(22)(7) and PULSE_GEN) xor GlobalCtrl(54)(7);
P_OUT(184) <= (GlobalCtrl(23)(0) and PULSE_GEN) xor GlobalCtrl(55)(0);
P_OUT(185) <= (GlobalCtrl(23)(1) and PULSE_GEN) xor GlobalCtrl(55)(1);
P_OUT(186) <= (GlobalCtrl(23)(2) and PULSE_GEN) xor GlobalCtrl(55)(2);
P_OUT(187) <= (GlobalCtrl(23)(3) and PULSE_GEN) xor GlobalCtrl(55)(3);
P_OUT(188) <= (GlobalCtrl(23)(4) and PULSE_GEN) xor GlobalCtrl(55)(4);
P_OUT(189) <= (GlobalCtrl(23)(5) and PULSE_GEN) xor GlobalCtrl(55)(5);
P_OUT(190) <= (GlobalCtrl(23)(6) and PULSE_GEN) xor GlobalCtrl(55)(6);
P_OUT(191) <= (GlobalCtrl(23)(7) and PULSE_GEN) xor GlobalCtrl(55)(7);
P_OUT(192) <= (GlobalCtrl(24)(0) and PULSE_GEN) xor GlobalCtrl(56)(0);
P_OUT(193) <= (GlobalCtrl(24)(1) and PULSE_GEN) xor GlobalCtrl(56)(1);
P_OUT(194) <= (GlobalCtrl(24)(2) and PULSE_GEN) xor GlobalCtrl(56)(2);
P_OUT(195) <= (GlobalCtrl(24)(3) and PULSE_GEN) xor GlobalCtrl(56)(3);
P_OUT(196) <= (GlobalCtrl(24)(4) and PULSE_GEN) xor GlobalCtrl(56)(4);
P_OUT(197) <= (GlobalCtrl(24)(5) and PULSE_GEN) xor GlobalCtrl(56)(5);
P_OUT(198) <= (GlobalCtrl(24)(6) and PULSE_GEN) xor GlobalCtrl(56)(6);
P_OUT(199) <= (GlobalCtrl(24)(7) and PULSE_GEN) xor GlobalCtrl(56)(7);
P_OUT(200) <= (GlobalCtrl(25)(0) and PULSE_GEN) xor GlobalCtrl(57)(0);
P_OUT(201) <= (GlobalCtrl(25)(1) and PULSE_GEN) xor GlobalCtrl(57)(1);
P_OUT(202) <= (GlobalCtrl(25)(2) and PULSE_GEN) xor GlobalCtrl(57)(2);
P_OUT(203) <= (GlobalCtrl(25)(3) and PULSE_GEN) xor GlobalCtrl(57)(3);
P_OUT(204) <= (GlobalCtrl(25)(4) and PULSE_GEN) xor GlobalCtrl(57)(4);
P_OUT(205) <= (GlobalCtrl(25)(5) and PULSE_GEN) xor GlobalCtrl(57)(5);
P_OUT(206) <= (GlobalCtrl(25)(6) and PULSE_GEN) xor GlobalCtrl(57)(6);
P_OUT(207) <= (GlobalCtrl(25)(7) and PULSE_GEN) xor GlobalCtrl(57)(7);
P_OUT(208) <= (GlobalCtrl(26)(0) and PULSE_GEN) xor GlobalCtrl(58)(0);
P_OUT(209) <= (GlobalCtrl(26)(1) and PULSE_GEN) xor GlobalCtrl(58)(1);
P_OUT(210) <= (GlobalCtrl(26)(2) and PULSE_GEN) xor GlobalCtrl(58)(2);
P_OUT(211) <= (GlobalCtrl(26)(3) and PULSE_GEN) xor GlobalCtrl(58)(3);
P_OUT(212) <= (GlobalCtrl(26)(4) and PULSE_GEN) xor GlobalCtrl(58)(4);
P_OUT(213) <= (GlobalCtrl(26)(5) and PULSE_GEN) xor GlobalCtrl(58)(5);
P_OUT(214) <= (GlobalCtrl(26)(6) and PULSE_GEN) xor GlobalCtrl(58)(6);
P_OUT(215) <= (GlobalCtrl(26)(7) and PULSE_GEN) xor GlobalCtrl(58)(7);
P_OUT(216) <= (GlobalCtrl(27)(0) and PULSE_GEN) xor GlobalCtrl(59)(0);
P_OUT(217) <= (GlobalCtrl(27)(1) and PULSE_GEN) xor GlobalCtrl(59)(1);
P_OUT(218) <= (GlobalCtrl(27)(2) and PULSE_GEN) xor GlobalCtrl(59)(2);
P_OUT(219) <= (GlobalCtrl(27)(3) and PULSE_GEN) xor GlobalCtrl(59)(3);
P_OUT(220) <= (GlobalCtrl(27)(4) and PULSE_GEN) xor GlobalCtrl(59)(4);
P_OUT(221) <= (GlobalCtrl(27)(5) and PULSE_GEN) xor GlobalCtrl(59)(5);
P_OUT(222) <= (GlobalCtrl(27)(6) and PULSE_GEN) xor GlobalCtrl(59)(6);
P_OUT(223) <= (GlobalCtrl(27)(7) and PULSE_GEN) xor GlobalCtrl(59)(7);
P_OUT(224) <= (GlobalCtrl(28)(0) and PULSE_GEN) xor GlobalCtrl(60)(0);
P_OUT(225) <= (GlobalCtrl(28)(1) and PULSE_GEN) xor GlobalCtrl(60)(1);
P_OUT(226) <= (GlobalCtrl(28)(2) and PULSE_GEN) xor GlobalCtrl(60)(2);
P_OUT(227) <= (GlobalCtrl(28)(3) and PULSE_GEN) xor GlobalCtrl(60)(3);
P_OUT(228) <= (GlobalCtrl(28)(4) and PULSE_GEN) xor GlobalCtrl(60)(4);
```

```
P_OUT(229) <= (GlobalCtrl(28)(5) and PULSE_GEN) xor GlobalCtrl(60)(5);
P_OUT(230) <= (GlobalCtrl(28)(6) and PULSE_GEN) xor GlobalCtrl(60)(6);
P_OUT(231) <= (GlobalCtrl(28)(7) and PULSE_GEN) xor GlobalCtrl(60)(7);
P_OUT(232) <= (GlobalCtrl(29)(0) and PULSE_GEN) xor GlobalCtrl(61)(0);
P_OUT(233) <= (GlobalCtrl(29)(1) and PULSE_GEN) xor GlobalCtrl(61)(1);
P_OUT(234) <= (GlobalCtrl(29)(2) and PULSE_GEN) xor GlobalCtrl(61)(2);
P_OUT(235) <= (GlobalCtrl(29)(3) and PULSE_GEN) xor GlobalCtrl(61)(3);
P_OUT(236) <= (GlobalCtrl(29)(4) and PULSE_GEN) xor GlobalCtrl(61)(4);
P_OUT(237) <= (GlobalCtrl(29)(5) and PULSE_GEN) xor GlobalCtrl(61)(5);
P_OUT(238) <= (GlobalCtrl(29)(6) and PULSE_GEN) xor GlobalCtrl(61)(6);
P_OUT(239) <= (GlobalCtrl(29)(7) and PULSE_GEN) xor GlobalCtrl(61)(7);
P_OUT(240) <= (GlobalCtrl(30)(0) and PULSE_GEN) xor GlobalCtrl(62)(0);
P_OUT(241) <= (GlobalCtrl(30)(1) and PULSE_GEN) xor GlobalCtrl(62)(1);
P_OUT(242) <= (GlobalCtrl(30)(2) and PULSE_GEN) xor GlobalCtrl(62)(2);
P_OUT(243) <= (GlobalCtrl(30)(3) and PULSE_GEN) xor GlobalCtrl(62)(3);
P_OUT(244) <= (GlobalCtrl(30)(4) and PULSE_GEN) xor GlobalCtrl(62)(4);
P_OUT(245) <= (GlobalCtrl(30)(5) and PULSE_GEN) xor GlobalCtrl(62)(5);
P_OUT(246) <= (GlobalCtrl(30)(6) and PULSE_GEN) xor GlobalCtrl(62)(6);
P_OUT(247) <= (GlobalCtrl(30)(7) and PULSE_GEN) xor GlobalCtrl(62)(7);
P_OUT(248) <= (GlobalCtrl(31)(0) and PULSE_GEN) xor GlobalCtrl(63)(0);
P_OUT(249) <= (GlobalCtrl(31)(1) and PULSE_GEN) xor GlobalCtrl(63)(1);
P_OUT(250) <= (GlobalCtrl(31)(2) and PULSE_GEN) xor GlobalCtrl(63)(2);
P_OUT(251) <= (GlobalCtrl(31)(3) and PULSE_GEN) xor GlobalCtrl(63)(3);
P_OUT(252) <= (GlobalCtrl(31)(4) and PULSE_GEN) xor GlobalCtrl(63)(4);
P_OUT(253) <= (GlobalCtrl(31)(5) and PULSE_GEN) xor GlobalCtrl(63)(5);
P_OUT(254) <= (GlobalCtrl(31)(6) and PULSE_GEN) xor GlobalCtrl(63)(6);
P_OUT(255) <= (GlobalCtrl(31)(7) and PULSE_GEN) xor GlobalCtrl(63)(7);


end circuit;
```

Arquivo 3 - DAC_Control.vhdl

```
--
**********************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:             LHCb SciFi Upgrade
-- Model:               Signal Injector Prototype
-- FPGA Proj. Name:     INJ_PROT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Ulisses Carneiro (ulissestv@gmail.com)
-- Date:                26-07-2015
-- ------------------------------------------------------------------------
-- Module:              DAC_Control
-- Description:         Controller for DACs over the Injector Board
--
--
**********************************************************************

-- ########################################################################
-- Revision History:
--    Date     Author            Rev.    Comments
-- 26 Jul 15  Ulisses Carneiro  0001    Project and file creation
-- ########################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

ENTITY DAC_Control IS
 PORT(
  --Pulses Input/Output:
  SCLK         : IN     std_logic;                    --Serial Clock (1MHz)

  --Digital to Analog Converter Interface:
  DAC_LDAC     : OUT    unsigned(1 to 2);             --DAC LDAC# IN
  DAC_CLK      : OUT    unsigned(1 to 2);             --DAC CLK IN
  DAC_IN       : OUT    unsigned(1 to 4);             --DAC DATA IN
  DAC_CSser    : OUT    unsigned(1 to 4);             --DAC CS# IN
  DAC_CStcp    : OUT    std_logic;                    --DAC CS# IN
  DAC_CShcp    : OUT    std_logic;                    --DAC CS# IN
  DAC_CSrst    : OUT    std_logic;                    --DAC CS# IN
  Debug1       : OUT    std_logic;                    --DAC CS# IN
  Debug2       : OUT    std_logic;                    --DAC CS# IN
  Debug3       : OUT    std_logic;                    --DAC CS# IN
  Debug4       : OUT    std_logic;                    --DAC CS# IN

 --Control:
  ControlData : IN     SubArray;                     --Ch 01 Dac Value

  TX_ON        : OUT    std_logic                     --DAC DATA TX IS ON

 );

END DAC_Control;

architecture dac_ctl of DAC_Control is
```

```
signal TX1_WORD:unsigned(15 downto 0);
signal TX2_WORD:unsigned(15 downto 0);
signal TX3_WORD:unsigned(15 downto 0);
signal TX4_WORD:unsigned(15 downto 0);
signal next_TX1_WORD:unsigned(15 downto 0);
signal next_TX2_WORD:unsigned(15 downto 0);
signal next_TX3_WORD:unsigned(15 downto 0);
signal next_TX4_WORD:unsigned(15 downto 0);
signal CS_WORD:unsigned(7 downto 0);
signal next_CS_WORD:unsigned(7 downto 0);
signal DECODED_CS:unsigned(7 downto 0);
signal DOUT1:std_logic;
signal DOUT2:std_logic;
signal DOUT3:std_logic;
signal DOUT4:std_logic;
signal CS_LOW:std_logic;
signal CS_HI:std_logic;
signal CS_SER1:std_logic;
signal SCLK_EN:std_logic;
signal CSCLK_EN:std_logic;
signal TX_State:std_logic;

signal TXcount:INTEGER range 0 to 31:=0;
signal WordCounter:unsigned(5 downto 0):="000000";
signal next_TXcount:INTEGER range 0 to 31:=0;
signal next_WordCounter:unsigned(5 downto 0):="000000";

begin
TX_State <= '0' when WordCounter = 0 else
            '1';

DECODED_CS <= "00000001" when WordCounter(5 downto 3) = 0 else
              "00000010" when WordCounter(5 downto 3) = 1 else
              "00000100" when WordCounter(5 downto 3) = 2 else
              "00001000" when WordCounter(5 downto 3) = 3 else
              "00010000" when WordCounter(5 downto 3) = 4 else
              "00100000" when WordCounter(5 downto 3) = 5 else
              "01000000" when WordCounter(5 downto 3) = 6 else
              "10000000";

Debug1 <= CS_SER1;
Debug2 <= CS_HI;
Debug3 <= SCLK and CSCLK_EN;
Debug4 <= not CS_LOW;

DAC_CLK(1) <= SCLK and SCLK_EN;
DAC_CLK(2) <= SCLK and SCLK_EN;
DAC_CSser(1) <= CS_SER1;
DAC_CSser(2) <= CS_SER1;
DAC_CSser(3) <= CS_SER1;
DAC_CSser(4) <= CS_SER1;
DAC_CStcp <= CS_HI;
DAC_CShcp <= SCLK and CSCLK_EN;
DAC_CSrst <= not CS_LOW;

DAC_IN(1) <= DOUT1;
DAC_IN(2) <= DOUT2;
DAC_IN(3) <= DOUT3;
DAC_IN(4) <= DOUT4;
DAC_LDAC(1) <= '1'     ;             --Hardware LDAC (Update all DACs)
DAC_LDAC(2) <= '1'     ;             --Hardware LDAC (Update all DACs)
TX_ON <= TX_State;
```

```
Transmitter:PROCESS(TXcount)
begin

case TXcount is
when 0 =>
  CS_LOW <= '0';
  CS_HI <= '0';
  DOUT1 <= '0';
  DOUT2 <= '0';
  DOUT3 <= '0';
  DOUT4 <= '0';
  SCLK_EN <= '0';
  CS_SER1 <= '0';
  CSCLK_EN <= '0';
  next_TXcount <= 1;
  next_WordCounter <= WordCounter;
  next_TX1_WORD <= "00"& WordCounter(2 downto 0) &"110"& Con-
trolData(conv_integer(WordCounter)+000);
  next_TX2_WORD <= "00"& WordCounter(2 downto 0) &"110"& Con-
trolData(conv_integer(WordCounter)+064);
  next_TX3_WORD <= "00"& WordCounter(2 downto 0) &"110"& Con-
trolData(conv_integer(WordCounter)+128);
  next_TX4_WORD <= "00"& WordCounter(2 downto 0) &"110"& Con-
trolData(conv_integer(WordCounter)+192);
  next_CS_WORD <= DECODED_CS;
when 1 =>
  CS_LOW <= '1';
  CS_HI <= '0';
  DOUT1 <= '0';
  DOUT2 <= '0';
  DOUT3 <= '0';
  DOUT4 <= '0';
  SCLK_EN <= '0';
  CS_SER1 <= '0';
  CSCLK_EN <= '0';
  next_TXcount <= 2;
  next_WordCounter <= WordCounter;
  next_TX1_WORD <= TX1_WORD;
  next_TX2_WORD <= TX2_WORD;
  next_TX3_WORD <= TX3_WORD;
  next_TX4_WORD <= TX4_WORD;
  next_CS_WORD <= CS_WORD;
when 2 =>
  CS_LOW <= '0';
  CS_HI <= '0';
  DOUT1 <= TX1_WORD(15);
  DOUT2 <= TX2_WORD(15);
  DOUT3 <= TX3_WORD(15);
  DOUT4 <= TX4_WORD(15);
  SCLK_EN <= '1';
  CS_SER1 <= '0';
  CSCLK_EN <= '0';
  next_TXcount <= 3;
  next_WordCounter <= WordCounter;
  next_TX1_WORD <= TX1_WORD;
  next_TX2_WORD <= TX2_WORD;
  next_TX3_WORD <= TX3_WORD;
  next_TX4_WORD <= TX4_WORD;
  next_CS_WORD <= CS_WORD;
when 3 =>
  CS_LOW <= '0';
  CS_HI <= '0';
  DOUT1 <= TX1_WORD(14);
```

```
    DOUT2 <= TX2_WORD(14);
    DOUT3 <= TX3_WORD(14);
    DOUT4 <= TX4_WORD(14);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 4;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 4 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(13);
    DOUT2 <= TX2_WORD(13);
    DOUT3 <= TX3_WORD(13);
    DOUT4 <= TX4_WORD(13);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 5;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 5 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(12);
    DOUT2 <= TX2_WORD(12);
    DOUT3 <= TX3_WORD(12);
    DOUT4 <= TX4_WORD(12);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 6;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 6 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(11);
    DOUT2 <= TX2_WORD(11);
    DOUT3 <= TX3_WORD(11);
    DOUT4 <= TX4_WORD(11);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 7;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
```

```
    next_CS_WORD <= CS_WORD;
when 7 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(10);
    DOUT2 <= TX2_WORD(10);
    DOUT3 <= TX3_WORD(10);
    DOUT4 <= TX4_WORD(10);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 8;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
when 8 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(09);
    DOUT2 <= TX2_WORD(09);
    DOUT3 <= TX3_WORD(09);
    DOUT4 <= TX4_WORD(09);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 9;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
when 9 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(08);
    DOUT2 <= TX2_WORD(08);
    DOUT3 <= TX3_WORD(08);
    DOUT4 <= TX4_WORD(08);
    SCLK_EN <= '1';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 10;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
when 10 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(07);
    DOUT2 <= TX2_WORD(07);
    DOUT3 <= TX3_WORD(07);
    DOUT4 <= TX4_WORD(07);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(7);
    CSCLK_EN <= '1';
    next_TXcount <= 11;
```

```
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 11 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(06);
    DOUT2 <= TX2_WORD(06);
    DOUT3 <= TX3_WORD(06);
    DOUT4 <= TX4_WORD(06);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(6);
    CSCLK_EN <= '1';
    next_TXcount <= 12;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 12 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(05);
    DOUT2 <= TX2_WORD(05);
    DOUT3 <= TX3_WORD(05);
    DOUT4 <= TX4_WORD(05);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(5);
    CSCLK_EN <= '1';
    next_TXcount <= 13;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 13 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(04);
    DOUT2 <= TX2_WORD(04);
    DOUT3 <= TX3_WORD(04);
    DOUT4 <= TX4_WORD(04);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(4);
    CSCLK_EN <= '1';
    next_TXcount <= 14;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 14 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(03);
    DOUT2 <= TX2_WORD(03);
    DOUT3 <= TX3_WORD(03);
```

```
    DOUT4 <= TX4_WORD(03);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(3);
    CSCLK_EN <= '1';
    next_TXcount <= 15;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 15 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(02);
    DOUT2 <= TX2_WORD(02);
    DOUT3 <= TX3_WORD(02);
    DOUT4 <= TX4_WORD(02);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(2);
    CSCLK_EN <= '1';
    next_TXcount <= 16;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 16 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(01);
    DOUT2 <= TX2_WORD(01);
    DOUT3 <= TX3_WORD(01);
    DOUT4 <= TX4_WORD(01);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(1);
    CSCLK_EN <= '1';
    next_TXcount <= 17;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 17 =>
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= TX1_WORD(00);
    DOUT2 <= TX2_WORD(00);
    DOUT3 <= TX3_WORD(00);
    DOUT4 <= TX4_WORD(00);
    SCLK_EN <= '1';
    CS_SER1 <= CS_WORD(0);
    CSCLK_EN <= '1';
    next_TXcount <= 18;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 18 =>
```

```
    CS_LOW <= '0';
    CS_HI <= '0';
    DOUT1 <= '0';
    DOUT2 <= '0';
    DOUT3 <= '0';
    DOUT4 <= '0';
    SCLK_EN <= '0';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 19;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 19 =>
    CS_LOW <= '0';
    CS_HI <= '1';
    DOUT1 <= '0';
    DOUT2 <= '0';
    DOUT3 <= '0';
    DOUT4 <= '0';
    SCLK_EN <= '0';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 20;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 20 =>
    CS_LOW <= '0';
    CS_HI <= '1';
    DOUT1 <= '0';
    DOUT2 <= '0';
    DOUT3 <= '0';
    DOUT4 <= '0';
    SCLK_EN <= '0';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    next_TXcount <= 21;
    next_WordCounter <= WordCounter;
    next_TX1_WORD <= TX1_WORD;
    next_TX2_WORD <= TX2_WORD;
    next_TX3_WORD <= TX3_WORD;
    next_TX4_WORD <= TX4_WORD;
    next_CS_WORD <= CS_WORD;
  when 21 =>
    CS_LOW <= '0';
    CS_HI <= '1';
    DOUT1 <= '0';
    DOUT2 <= '0';
    DOUT3 <= '0';
    DOUT4 <= '0';
    SCLK_EN <= '0';
    CS_SER1 <= '0';
    CSCLK_EN <= '0';
    if (WordCounter = conv_unsigned(63,6)) then
      next_TXcount <= 22;
    else
```

```
    next_TXcount <= 23;
  end if;
  next_WordCounter <= WordCounter;
  next_TX1_WORD <= TX1_WORD;
  next_TX2_WORD <= TX2_WORD;
  next_TX3_WORD <= TX3_WORD;
  next_TX4_WORD <= TX4_WORD;
  next_CS_WORD <= CS_WORD;
when 22 =>
  CS_LOW <= '0';
  CS_HI <= '1';
  DOUT1 <= '0';
  DOUT2 <= '0';
  DOUT3 <= '0';
  DOUT4 <= '0';
  SCLK_EN <= '0';
  CS_SER1 <= '0';
  CSCLK_EN <= '0';
  next_TXcount <= 23;
  next_WordCounter <= WordCounter;
  next_TX1_WORD <= TX1_WORD;
  next_TX2_WORD <= TX2_WORD;
  next_TX3_WORD <= TX3_WORD;
  next_TX4_WORD <= TX4_WORD;
  next_CS_WORD <= CS_WORD;
when 23 =>
  CS_LOW <= '0';
  CS_HI <= '1';
  DOUT1 <= '0';
  DOUT2 <= '0';
  DOUT3 <= '0';
  DOUT4 <= '0';
  SCLK_EN <= '0';
  CS_SER1 <= '0';
  CSCLK_EN <= '0';
  next_WordCounter <= WordCounter + conv_unsigned(1,6);
  next_TXcount <= 0;
  next_TX1_WORD <= TX1_WORD;
  next_TX2_WORD <= TX2_WORD;
  next_TX3_WORD <= TX3_WORD;
  next_TX4_WORD <= TX4_WORD;
  next_CS_WORD <= DECODED_CS;
when others =>
  next_TXcount <= 0;
end case;
end process;

StMachineSeq:PROCESS(SCLK)
begin
if SCLK'event and SCLK='0'then
  TXcount <= next_TXcount;
  WordCounter <= next_WordCounter;
  TX1_WORD <= next_TX1_WORD;
  TX2_WORD <= next_TX2_WORD;
  TX3_WORD <= next_TX3_WORD;
  TX4_WORD <= next_TX4_WORD;
  CS_WORD <= next_CS_WORD;
end if;
end process;

end dac_ctl;
```

Arquivo 4 - RegMemory.vhdl

```
-- ******************************************************************************
-- Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:            LHCb SciFi Upgrade
-- Model:              Signal Injector Prototype
-- FPGA Proj. Name:    INJ_PROT
-- Device:             ALTERA EP4CE10F17C8
-- Author:             Ulisses Carneiro (ulissestv@gmail.com)
-- Date:               26-07-2015
-- ------------------------------------------------------------------------------
-- Module:             RegMemory
-- Description:        Register Memory for Injector Board FPGA
--
-- ******************************************************************************

-- ##############################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 26 Jul 15  Ulisses Carneiro  0001   Project and file creation
-- ##############################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

entity RegMemory is

  port (
    clk      : in std_logic;
       Msel     : in std_logic;
    Aaddr    : in unsigned(15 downto 0 );
    AdataIn  : in unsigned(7 downto 0 );
    AwriteEn : in std_logic;
    AdataOut : buffer unsigned(7 downto 0);
    Baddr    : in unsigned(15 downto 0 );
    BdataIn  : in unsigned(7 downto 0 );
    BwriteEn : in std_logic;
    BdataOut : buffer unsigned(7 downto 0);
    myReg    : buffer MainArray;
    FWver    : in unsigned(7 downto 0 )
  );
end entity;


architecture rtl of RegMemory is

signal addr    :unsigned(15 downto 0 );
signal dataIn  :unsigned(7 downto 0 );
signal writeEn :std_logic;

begin

addr    <= Aaddr when Msel = '0' else
           Baddr;
dataIn  <= AdataIn when Msel = '0' else
           BdataIn;
```

```
writeEn <= AwriteEn when Msel = '0' else
           BwriteEn;

LoadARegToI2C:process(clk)
begin
if (clk'EVENT and clk = '1') then
  if (Aaddr = conv_unsigned(2303, 16)) then
  AdataOut <= FWver;
  else
  AdataOut <= myReg(conv_integer(Aaddr));
  end if;
end if;
end process;

LoadBRegToI2C:process(clk)
begin
if (clk'EVENT and clk = '1') then
  if (Baddr = conv_unsigned(2303, 16)) then
  BdataOut <= FWver;
  else
  BdataOut <= myReg(conv_integer(Baddr));
  end if;
end if;
end process;

LoadRegFromI2C:process(clk)
begin
if (clk'EVENT and clk = '1') then
  if (writeEn = '1') then
    myReg(conv_integer(addr)) <= dataIn;
  end if;
end if;
end process;

end;
```

Arquivo 5 - RegAdapter.vhdl

```
-- ***************************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:             LHCb SciFi Upgrade
-- Model:               Signal Injector Prototype
-- FPGA Proj. Name:     INJ_PROT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Ulisses Carneiro (ulissestv@gmail.com)
-- Date:                26-07-2015
-- ---------------------------------------------------------------------------
-- Module:              RegAdapter
-- Description:         Register Bank Adapter for Injector Board FPGA
-- ###########################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 26 Jul 15  Ulisses Carneiro  0001   Project and file creation
-- ###########################################################################
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;
LIBRARY work;
USE work.my_types_pkg.all;

entity RegAdapter is
  port (
    Scene     : in unsigned(1 downto 0);
    myReg     : in MainArray;
    ADCReg    : out SubArray;
    RiseReg   : out SubArray;
    FallReg   : out SubArray;
    SpareReg  : out SubArray;
    GlobalReg : out SubArray
  );
end entity;

architecture rtl of RegAdapter is
begin
process (myReg,Scene)
begin
if Scene = 0 then
  for c in 0 to 255 loop
    ADCReg(c)   <= myReg(00*256+c);
    RiseReg(c)  <= myReg(01*256+c);
    FallReg(c)  <= myReg(02*256+c);
    SpareReg(c) <= myReg(03*256+c);
  end loop;
else
  for c in 0 to 255 loop
    ADCReg(c)   <= myReg(04*256+c);
    RiseReg(c)  <= myReg(05*256+c);
    FallReg(c)  <= myReg(06*256+c);
    SpareReg(c) <= myReg(07*256+c);
  end loop;
end if;
for c in 0 to 255 loop
  GlobalReg(c)  <= myReg(08*256+c);
end loop;
end process;
end;
```

Arquivo 6 - i2cSlave.v

```
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// i2cSlave.v                                                   ////
////                                                              ////
//// This file is part of the i2cSlave opencores effort.          ////
//// <http://www.opencores.org/cores//>                           ////
////                                                              ////
//// Module Description:                                          ////
//// You will need to modify this file to implement your          ////
//// interface.                                                   ////
////                                                              ////
//// To Do:                                                       ////
////                                                              ////
//// Author(s):                                                   ////
//// - Steve Fielding, sfielding@base2designs.com                 ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
////                                                              ////
//// Copyright (C) 2008 Steve Fielding and OPENCORES.ORG          ////
////                                                              ////
//// This source file may be used and distributed without         ////
//// restriction provided that this copyright statement is not    ////
//// removed from the file and that any derivative work contains  ////
//// the original copyright notice and the associated disclaimer. ////
////                                                              ////
//// This source file is free software; you can redistribute it   ////
//// and/or modify it under the terms of the GNU Lesser General   ////
//// Public License as published by the Free Software Foundation; ////
//// either version 2.1 of the License, or (at your option) any   ////
//// later version.                                               ////
////                                                              ////
//// This source is distributed in the hope that it will be       ////
//// useful, but WITHOUT ANY WARRANTY; without even the implied   ////
//// warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR      ////
//// PURPOSE. See the GNU Lesser General Public License for more  ////
//// details.                                                     ////
////                                                              ////
//// You should have received a copy of the GNU Lesser General    ////
//// Public License along with this source; if not, download it   ////
//// from <http://www.opencores.org/lgpl.shtml>                   ////
////                                                              ////
//////////////////////////////////////////////////////////////////////
`include "i2cSlave_define.v"

module i2cSlave (
  clk,
  rst,
  sda,
  scl,
  addr,
  dataIn,
  writeEn,
  dataOut
);

input clk;
input rst;

inout sda;
input scl;
```

```
output [15:0] addr;
output [7:0] dataIn;
output writeEn;
input [7:0] dataOut;

// local wires and regs
reg sdaDeb;
reg sclDeb;
reg [`DEB_I2C_LEN-1:0] sdaPipe;
reg [`DEB_I2C_LEN-1:0] sclPipe;
reg [`SCL_DEL_LEN-1:0] sclDelayed;
reg [`SDA_DEL_LEN-1:0] sdaDelayed;
reg [1:0] startStopDetState;
wire clearStartStopDet;
wire sdaOut;
wire sdaIn;
wire [15:0] regAddr;
wire [7:0] dataToRegIF;
wire writeEn;
wire [7:0] dataFromRegIF;
reg [1:0] rstPipe;
wire rstSyncToClk;
reg startEdgeDet;

assign sda = (sdaOut == 1'b0) ? 1'b0 : 1'bz;
assign sdaIn = sda;
assign addr=regAddr;
assign dataIn=dataToRegIF;
assign dataFromRegIF=dataOut;

// sync rst rsing edge to clk
always @(posedge clk) begin
  if (rst == 1'b1)
    rstPipe <= 2'b11;
  else
    rstPipe <= {rstPipe[0], 1'b0};
end

assign rstSyncToClk = rstPipe[1];

// debounce sda and scl
always @(posedge clk) begin
  if (rstSyncToClk == 1'b1) begin
    sdaPipe <= {`DEB_I2C_LEN{1'b1}};
    sdaDeb <= 1'b1;
    sclPipe <= {`DEB_I2C_LEN{1'b1}};
    sclDeb <= 1'b1;
  end
  else begin
    sdaPipe <= {sdaPipe[`DEB_I2C_LEN-2:0], sdaIn};
    sclPipe <= {sclPipe[`DEB_I2C_LEN-2:0], scl};
    if (&sclPipe[`DEB_I2C_LEN-1:1] == 1'b1)
      sclDeb <= 1'b1;
    else if (|sclPipe[`DEB_I2C_LEN-1:1] == 1'b0)
      sclDeb <= 1'b0;
    if (&sdaPipe[`DEB_I2C_LEN-1:1] == 1'b1)
      sdaDeb <= 1'b1;
    else if (|sdaPipe[`DEB_I2C_LEN-1:1] == 1'b0)
      sdaDeb <= 1'b0;
  end
end
```

```
// delay scl and sda
// sclDelayed is used as a delayed sampling clock
// sdaDelayed is only used for start stop detection
// Because sda hold time from scl falling is 0nS
// sda must be delayed with respect to scl to avoid incorrect
// detection of start/stop at scl falling edge.
always @(posedge clk) begin
  if (rstSyncToClk == 1'b1) begin
    sclDelayed <= {`SCL_DEL_LEN{1'b1}};
    sdaDelayed <= {`SDA_DEL_LEN{1'b1}};
  end
  else begin
    sclDelayed <= {sclDelayed[`SCL_DEL_LEN-2:0], sclDeb};
    sdaDelayed <= {sdaDelayed[`SDA_DEL_LEN-2:0], sdaDeb};
  end
end

// start stop detection
always @(posedge clk) begin
  if (rstSyncToClk == 1'b1) begin
    startStopDetState <= `NULL_DET;
    startEdgeDet <= 1'b0;
  end
  else begin
    if (sclDeb == 1'b1 && sdaDelayed[`SDA_DEL_LEN-2] == 1'b0 && sdaDe-
layed[`SDA_DEL_LEN-1] == 1'b1)
      startEdgeDet <= 1'b1;
    else
      startEdgeDet <= 1'b0;
    if (clearStartStopDet == 1'b1)
      startStopDetState <= `NULL_DET;
    else if (sclDeb == 1'b1) begin
      if (sdaDelayed[`SDA_DEL_LEN-2] == 1'b1 && sdaDelayed[`SDA_DEL_LEN-1]
== 1'b0)
        startStopDetState <= `STOP_DET;
      else if (sdaDelayed[`SDA_DEL_LEN-2] == 1'b0 && sdaDe-
layed[`SDA_DEL_LEN-1] == 1'b1)
        startStopDetState <= `START_DET;
    end
  end
end

serialInterface u_serialInterface (
  .clk(clk),
  .rst(rstSyncToClk | startEdgeDet),
  .dataIn(dataFromRegIF),
  .dataOut(dataToRegIF),
  .writeEn(writeEn),
  .regAddr(regAddr),
  .scl(sclDelayed[`SCL_DEL_LEN-1]),
  .sdaIn(sdaDeb),
  .sdaOut(sdaOut),
  .startStopDetState(startStopDetState),
  .clearStartStopDet(clearStartStopDet)
);
endmodule
```

Arquivo 7 - SclkDivider.vhdl

```
-- ******************************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:             LHCb SciFi Upgrade
-- Model:               Charge Injector
-- FPGA Proj. Name:     INJ_PROT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Ulisses Carneiro (ulissestv@gmail.com)
-- Date:                28-03-2017
-- ------------------------------------------------------------------------------
-- Module:              SclkDivider
-- Description:
-- ******************************************************************************

-- ##############################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 28 Mar 17  Ulisses Carneiro  0001   Project and file creation
-- ##############################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

ENTITY SclkDivider IS
 PORT(
  --Pulses Input/Output:
  LCLK        : IN     std_logic;            --Local Clock (50MHz)
  SCLK        : OUT    std_logic             --Serial Clock for DAC Control

 );
END SclkDivider;

architecture circuit of SclkDivider is

begin

SerialClock:process(LCLK)
variable count:integer range 0 to 40;
begin
  if LCLK'event and LCLK='1' then
    count:=count+1;
       if count < 20 then
      SCLK<='0';
    elsif count < 40 then
      SCLK<='1';
    else count:=0;
    end if;
  end if;
end process;

end circuit;
```

Arquivo 8 - eLinkDecoder.vhdl

```
-- ************************************************************************
-- Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:            LHCb SciFi Upgrade
-- Model:              Signal Injector Prototype
-- FPGA Proj. Name:    INJ_PROT
-- Device:             ALTERA EP4CE10F17C8
-- Author:             Ulisses Carneiro (ulissestv@gmail.com)
-- Date:               26-07-2015
-- ------------------------------------------------------------------------
-- Module:             eLinkDecoder
-- Description:        Decoder for eLink signals at Injector Board FPGA
--
-- ************************************************************************

-- ########################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 26 Jul 15  Ulisses Carneiro  0001   Project and file creation
-- ########################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

entity eLinkDecoder is

  port (
    clk       : in std_logic;
    eClk      : in std_logic;
    eLinkA    : in std_logic;
    eLinkB    : in std_logic;
    GlobalReg : in SubArray;
       Scene     : out unsigned(1 downto 0)
  );
end entity;

architecture rtl of eLinkDecoder is

signal eLink_decoded :unsigned (1 downto 0);
signal SceneSeq      :unsigned (1 downto 0);

begin

DDRdecoder:process (eClk,eLinkA,eLinkB)
begin

if eClk'event and eClk='0' then
  eLink_decoded(0) <= eLinkA;
end if;

if eClk'event and eClk='1' then
  eLink_decoded(1) <= eLinkA;
end if;
end process;
```

```
SceneSelector:process (clk,eLink_decoded,SceneSeq,GlobalReg)
begin

if GlobalReg(128)(7)='0' then
  Scene <= GlobalReg(128)(1 downto 0);
else
  if GlobalReg(128)(6)='0' then
    Scene <= eLink_decoded(1 downto 0);
  else
    Scene <= SceneSeq;
  end if;
end if;
end process;

Sequential:process (clk,GlobalReg)
variable count : unsigned (3 downto 0);
begin
if clk'event and clk='1' then
  count:=count+1;
  if count = GlobalReg(128)(5 downto 2) then
    SceneSeq <= SceneSeq+'1';
       count:="0000";
  end if;
end if;
end process;

end rtl;
```

Arquivo 9 - LedStatus.vhdl

```
-- ******************************************************************************
-- Company:             CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:             LHCb SciFi Upgrade
-- Model:               Signal Injector Prototype
-- FPGA Proj. Name:     INJ_PROT
-- Device:              ALTERA EP4CE10F17C8
-- Author:              Ulisses Carneiro (ulissestv@gmail.com)
-- Date:                26-07-2015
-- ------------------------------------------------------------------------------
-- Module:              LedStatus
-- Description:         Decoder for LEDs of the Injector Board
--
-- ******************************************************************************

-- ##############################################################################
-- Revision History:
--    Date     Author            Rev.    Comments
-- 26 Jul 15  Ulisses Carneiro  0001    Project and file creation
-- ##############################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

entity LedStatus is

  port (
    myReg    : in MainArray;
    Status1  : out std_logic;
    Status2  : out std_logic;
    Status3  : out std_logic;
    Status4  : out std_logic;
    Status5  : out std_logic;
    Status6  : out std_logic;
    Status7  : out std_logic;
    Status8  : out std_logic
  );
end entity;

architecture rtl of LedStatus is
begin

Status1  <= myReg(2192)(0);
Status2  <= myReg(2192)(1);
Status3  <= myReg(2192)(2);
Status4  <= myReg(2192)(3);
Status5  <= myReg(2192)(4);
Status6  <= myReg(2192)(5);
Status7  <= myReg(2192)(6);
Status8  <= myReg(2192)(7);

end;
```

Arquivo 10 - LedMasterI2C.vhdl

```
-- ***************************************************************************
-- Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:            LHCb SciFi Upgrade
-- Model:              Signal Injector Prototype
-- FPGA Proj. Name:    INJ_PROT
-- Device:             ALTERA EP4CE10F17C8
-- Author:             Ulisses Carneiro (ulissestv@gmail.com)
-- Date:               26-07-2015
-- ---------------------------------------------------------------------------
-- Module:             LedMasterI2C
-- Description:        I2C Master Interface to control the LED Driver of the
--                     Injector Board
-- ***************************************************************************

-- ###########################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 26 Jul 15  Ulisses Carneiro  0001   Project and file creation
-- ###########################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

entity LedMasterI2C is

  port (
    clk     : in std_logic;

    Led1    : in std_logic;
    Led2    : in std_logic;
    Led3    : in std_logic;
    Led4    : in std_logic;
    Led5    : in std_logic;
    Led6    : in std_logic;
    Led7    : in std_logic;
    Led8    : in std_logic;
    Led9    : in std_logic;
    Led10   : in std_logic;
    Led11   : in std_logic;
    Led12   : in std_logic;
    Led13   : in std_logic;
    Led14   : in std_logic;
    Led15   : in std_logic;
    Led16   : in std_logic;

    sda     : out std_logic;
    scl     : out std_logic;

       monitor : out unsigned (6 downto 0)
  );
end entity;


architecture rtl of LedMasterI2C is
```

```
-- Set PCA9685 Slave Address and Mode1 / Mode2 registers:
constant SlaveAddr      : unsigned(7 downto 1):="1010101";
constant Mode1          : unsigned(7 downto 0):="00100000";
constant Mode2          : unsigned(7 downto 0):="00010010";

-- Prescaler settings:
constant CLK_FREQ       : integer:= 40000; --kHz
constant I2Cfreq        : integer:= 400; --kHz

-- Internal Prescaler values (Don't modify here):
constant FullCicle      : integer:= (CLK_FREQ/I2Cfreq)-1;
constant HalfCicle      : integer:= ((CLK_FREQ/I2Cfreq)/2)-1;
constant QuarCicle      : integer:= (3*(CLK_FREQ/I2Cfreq)/4)-1;
constant SCL_RISE_TUNE  : integer:= 3;

-- SISt states
constant START1         : unsigned(2 downto 0):= "000";
constant START2         : unsigned(2 downto 0):= "001";
constant SEND_DATA      : unsigned(2 downto 0):= "010";
constant GET_ACK        : unsigned(2 downto 0):= "011";
constant PAUSE          : unsigned(2 downto 0):= "100";
constant STOP1          : unsigned(2 downto 0):= "101";
constant STOP2          : unsigned(2 downto 0):= "110";

-- SCL states
constant SCL_HI         : unsigned(1 downto 0):= "00";
constant SCL_LOW        : unsigned(1 downto 0):= "01";
constant SCL_OSC        : unsigned(1 downto 0):= "10";

-- PCA9685 reg. states
constant LEDn_ON_L      : unsigned(1 downto 0):= "11";
constant LEDn_ON_H      : unsigned(1 downto 0):= "00";
constant LEDn_OFF_L     : unsigned(1 downto 0):= "01";
constant LEDn_OFF_H     : unsigned(1 downto 0):= "10";
constant LAST_REG       : integer:= 70;

signal CurrState_SISt   : unsigned(2 downto 0);
signal NextState_SISt   : unsigned(2 downto 0);

signal SCLstate         : unsigned(1 downto 0);
signal next_SCLstate    : unsigned(1 downto 0);

signal BitCount         : unsigned (3 downto 0);
signal ByteCount        : unsigned (6 downto 0);
signal SCLo             : std_logic;
signal SDAo             : std_logic;

signal next_BitCount    : unsigned (3 downto 0);
signal next_ByteCount   : unsigned (6 downto 0);
signal next_SDAo        : std_logic;

signal count0           : unsigned (6 downto 0);
signal sclClk           : std_logic;
signal sdaClk           : std_logic;

signal TXdata           : unsigned (8 downto 1);
signal next_TXdata      : unsigned (8 downto 1);

signal LEDon            : unsigned (16 downto 1);

begin
```

```
scl <= '0' when (SCLo = '0') else 'Z';
sda <= '0' when (SDAo = '0') else 'Z';

LEDon(1 ) <= LED1;
LEDon(2 ) <= LED2;
LEDon(3 ) <= LED3;
LEDon(4 ) <= LED4;
LEDon(5 ) <= LED5;
LEDon(6 ) <= LED6;
LEDon(7 ) <= LED7;
LEDon(8 ) <= LED8;
LEDon(9 ) <= LED9;
LEDon(10) <= LED10;
LEDon(11) <= LED11;
LEDon(12) <= LED12;
LEDon(13) <= LED13;
LEDon(14) <= LED14;
LEDon(15) <= LED15;
LEDon(16) <= LED16;

monitor <= ByteCount;

ClkDivider:process(clk)
begin
if clk'event and clk='1' then
  if count0 < conv_unsigned(FullCicle,7) then
    count0 <= count0 + conv_unsigned(01,7);
  else
    count0 <= conv_unsigned(00,7);
  end if;
end if;
end process;


StMachine:process (TXdata, BitCount, ByteCount, CurrState_SISt)
begin

case CurrState_SISt is
when START1 =>
  next_SDAo <= '1';
  next_SCLstate <= SCL_HI;
  next_BitCount <= conv_unsigned(8,4);
  next_ByteCount <= conv_unsigned(00,7);
  next_TXdata <= SlaveAddr & '0';
  NextState_SISt <= START2;
when START2 =>
  next_SDAo <= '0';
  next_SCLstate <= SCL_OSC;
  next_BitCount <= conv_unsigned(8,4);
  next_ByteCount <= conv_unsigned(00,7);
  NextState_SISt <= SEND_DATA;
when SEND_DATA =>
  next_SDAo <= TXdata(conv_integer(BitCount));
  next_SCLstate <= SCL_OSC;
  if (BitCount /= conv_unsigned(0,4)) then
    next_BitCount <= BitCount - conv_unsigned(1,4);
    NextState_SISt <= SEND_DATA;
  else
    NextState_SISt <= GET_ACK;
  end if;
when GET_ACK =>
  next_SDAo <= '1';
  next_SCLstate <= SCL_OSC;
```

```
  NextState_SISt <= PAUSE;
  next_ByteCount <= ByteCount + conv_unsigned(1,7);
when PAUSE =>
  next_SDAo <= '1';
  next_SCLstate <= SCL_LOW;
  next_BitCount <= conv_unsigned(8,4);
  if (ByteCount = conv_unsigned(01,7)) then
    next_TXdata <= Mode1;
  elsif (ByteCount = conv_unsigned(02,7)) then
    next_TXdata <= Mode2;
  elsif (ByteCount <= conv_unsigned(06,7)) then
    next_TXdata <= "00000000";
  elsif (ByteCount(1 downto 0) = LEDn_ON_H) then
    next_TXdata <= "00010000";
  elsif (ByteCount(1 downto 0) = LEDn_OFF_H) then
    next_TXdata <= "000" & LEDon(conv_integer(ByteCount(6 downto 2))-1) &
"0000";
  else
    next_TXdata <= "00000000";
  end if;
  if (ByteCount > conv_unsigned(LAST_REG,7)) then
    NextState_SISt <= STOP1;
  else
    NextState_SISt <= SEND_DATA;
  end if;
when STOP1 =>
  next_SDAo <= '0';
  next_SCLstate <= SCL_HI;
  NextState_SISt <= STOP2;
when STOP2 =>
  next_SDAo <= '1';
  next_SCLstate <= SCL_HI;
  NextState_SISt <= START1;
when others =>
  NextState_SISt <= START1;
end case;
end process;


-- Current State Logic (sequential):
NextStateMachine:process(clk,count0)
begin
if clk'event and clk='0' then
  if count0 = conv_unsigned(0,7) then
  CurrState_SISt <= NextState_SISt;
  ByteCount <= next_ByteCount;
  end if;
end if;
end process;

-- Registered outputs logic:
LoadData:process(clk,count0)
begin
if clk'event and clk='0' then
  if count0 = conv_unsigned(QuarCicle,7) then
  SDAo <= next_SDAo;
  SCLstate <= next_SCLstate;
  BitCount <= next_BitCount;
  TXdata <= next_TXdata;
  end if;
end if;
end process;
```

```
process(clk,count0)
begin
if clk'event and clk='0' then
  if count0 = conv_unsigned(SCL_RISE_TUNE,7) then
    if (SCLstate = SCL_LOW) then
      SCLo <= '0';
    else
      SCLo <= '1';
    end if;
  end if;
  if (count0 = conv_unsigned(HalfCicle,7)) then
    if (SCLstate = SCL_HI) then
      SCLo <= '1';
    else
      SCLo <= '0';
    end if;
  end if;
end if;
end process;


end rtl;
```

Arquivo 11 - UBSinterface.vhdl

```
-- ******************************************************************************
-- Company:            CENTRO BRASILEIRO DE PESQUISAS FISICAS - CBPF BRASIL
-- Project:            LHCb SciFi Upgrade
-- Model:              Signal Injector Prototype
-- FPGA Proj. Name:    INJ_PROT
-- Device:             ALTERA EP4CE10F17C8
-- Author:             Ulisses Carneiro (ulissestv@gmail.com)
-- Date:               26-07-2015
-- ------------------------------------------------------------------------------
-- Module:             UBSinterface
-- Description:        USB/FTDI interface for Injector Board FPGA
--
-- ******************************************************************************

-- ##############################################################################
-- Revision History:
--   Date     Author            Rev.   Comments
-- 26 Jul 15  Ulisses Carneiro  0001   Project and file creation
-- ##############################################################################

LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_misc.all;

LIBRARY work;
USE work.my_types_pkg.all;

entity USBinterface is

  port (
    clk       : in std_logic;
    USB_data  : inout unsigned(7 downto 0);
    USB_RXF   : in std_logic;
    USB_TXE   : in std_logic;
    USB_RD    : out std_logic;
    USB_WR    : out std_logic;

    addr      : out unsigned(15 downto 0 );
    dataIn    : out unsigned(7 downto 0 );
    writeEn   : out std_logic;
    dataOut   : in unsigned(7 downto 0)
  );
end entity;

architecture rtl of USBinterface is

begin


end rtl;
```

Apêndice E

Código C^{++} da Seção 3.9.1

```
000 #include <Wire.h>
001
002 int value;
003 byte received;
004 byte state;
005 byte SlaveAddress = 0x00;
006 byte next_write;
007 unsigned int TxSeq = 0x0000;
008 boolean halfByte=true;
009 boolean illegalInput=false;
010 unsigned int InitialAddr, NumberAddr, BufferLength;
011 boolean TxPaused, TxLastState = false;
012 byte CRCtx, CRCrx;
013
014 void setup()
015 {
016   Wire.begin();
017   Serial.begin(9600);
018   Serial.println();
019   Serial.println("----------------------------");
020   Serial.println("- LHCb UPGRADE TEST SYSTEM -");
021   Serial.println("- I2C ADAPTER FOR INJECTOR -");
022   Serial.println("----------------------------");
023   I2Cscan();
024   CommandList();
025 }
026
027 void loop()
028 {
029 byte UARTbufferIndex = (byte)Serial.available();
030
031 if(UARTbufferIndex>0x04) TxPaused = true;
032 if(UARTbufferIndex==0x00) TxPaused = false;
033
034 if(TxPaused==true&&TxLastState==false) {
035   Serial.write(0x13); // Flow Control - Pause TX
036   TxLastState=TxPaused;
037 }
038 if(TxPaused==false&&TxLastState==true) {
039   Serial.write(0x11); // Flow Control - Resume TX
040   TxLastState=TxPaused;
041 }
042
043 if(UARTbufferIndex>0)
044   {
045     byte inChar=(byte)Serial.read();  // Read next byte of UART FIFO
046     if(inChar==0x0D)
047     {
048       switch (state)
049       {
050         case 0x1:       // Case user press Enter on write mode step 1
(after enter Initial Reg Add)
051           InitiateI2C();
052           halfByte=true;
053           TxSeq = 0x00;
054           state=0x2;
055           Serial.println();
056           Serial.println("Enter data to TX (XXh):");
057           PrintRegIndex(TxSeq);
058           break;
059         case 0x3:       // Case user press Enter on read mode step 1
(after enter Initial Reg Add)
060           state=0x4;
```

```
061           Serial.println();
062           Serial.print("Number of addresses to read (XXh): ");
063           break;
064         case 0x2:       // Case user press Enter on write mode step 2
(after enter all data to TX)
065           Serial.println();
066           Serial.println();
067           TxCicle();
068           state = 0x0;   // Jump to idle state
069           break;
070         case 0x4:       // Case user press Enter on read mode step 2
(after enter number of regs to RX)
071           Serial.println();
072           ReadI2C(true);
073           state = 0x0;   // Jump to idle state
074           break;
075         case 0x5:
076           SlaveAddress = InitialAddr;
077           Serial.println();
078           Serial.print("INFO - Slave Address set to: 0x");
079           Serial.println(SlaveAddress, HEX);
080           state = 0x0;   // Jump to idle state
081           break;
082         default:
083           CommandList();
084       }
085       return;
086     }
087     if(state==0x2)        // If user press any key on write mode step 2
(enter Data to TX)
088     {
089       if(inChar==0x1B)
090       {
091         state=0x0;
092         AbortInfo();
093       }
094       byte converted = convert(inChar);
095       if(illegalInput) return;
096       Serial.print(converted, HEX);
097       if(halfByte)
098       {
099         next_write = convert(inChar);
100         halfByte=false;
101       } else {
102         next_write = next_write << 4;
103         next_write = next_write|converted;
104         Wire.write(next_write);
105         Serial.print(" ");
106         halfByte=true;
107         TxSeq++;
108       }
109       if (TxSeq == 0x100)
110       {
111         if (TxPaused==false){
112           Serial.write(0x13); // Flow Control - Pause TX
113           TxPaused = true;
114           TxLastState = true;
115         }
116         Serial.println();
117         Serial.println();
118         Serial.println("INFO - Partial data TX");
119         TxCicle();
120         ReadI2C(false);
```

```
121         if(CRCrx==CRCtx){
122           Serial.println("INFO - CRC check ok");
123         } else {
124           Serial.println("WARN - CRC check FAIL");
125         }
126         TxSeq=0x00;
127         InitialAddr=InitialAddr+0x0100;
128         InitiateI2C();
129       }
130       if ((TxSeq&0x000F)==0x0000&&halfByte==true)
131       {
132         PrintRegIndex(TxSeq);
133       }
134       return;
135     }
136     if(state==0x3||state==0x5||state==0x1)
137     {
138       if(inChar==0x1B)
139       {
140         state=0x0;
141         AbortInfo();
142       }
143       byte converted = convert(inChar);
144       if(illegalInput) return;
145       Serial.print(converted, HEX);
146       InitialAddr = InitialAddr << 4;
147       InitialAddr = InitialAddr|converted;
148       return;
149     }
150     if(state==0x4)
151     {
152       if(inChar==0x1B)
153       {
154         state=0x0;
155         AbortInfo();
156       }
157       byte converted = convert(inChar);
158       if(illegalInput) return;
159       Serial.print(converted, HEX);
160       NumberAddr = NumberAddr << 4;
161       NumberAddr = NumberAddr|converted;
162       return;
163     }
164     if(inChar==0x57||inChar==0x77)  // If receives a 'W': Go to write
mode (0x1)
165     {
166       if(state!=0x0) AbortInfo();
167       state = 0x1;
168       InitialAddr = 0x0000;
169       Serial.print("Write mode. Set initial address (XXXXh): ");
170       return;
171     }
172     if(inChar==0x52||inChar==0x72)  // If receives a 'R': Go to read
mode (0x3)
173     {
174       if(state!=0x0) AbortInfo();
175       state = 0x3;
176       InitialAddr = 0x0000;
177       NumberAddr = 0x00;
178       Serial.print("Read mode. Set initial address (XXXXh): ");
179       return;
180     }
```

```
181     if(inChar==0x41||inChar==0x61)  // If receives a 'A': Go to address
change mode (0x5)
182     {
183       if(state!=0x0) AbortInfo();
184       state = 0x5;
185       InitialAddr = 0x0000;
186       Serial.print("Enter slave address (XXh): ");
187       return;
188     }
189     if(inChar==0x53||inChar==0x73)
190     {
191       I2Cscan();
192     }
193   }
194 }
195
196 byte ComputeCRC(byte initial)
197 {
198   byte remainder = 0;
199   for (int n_byte = initial; n_byte < BufferLength; ++n_byte)
200   {
201     remainder ^= Wire.ReadMasterBuffer(n_byte);
202     for (uint8_t bit = 8; bit > 0; --bit)
203     {
204       if (remainder & 0x80)
205       {
206           remainder = (remainder << 1) ^ 0xD8;
207       }
208       else
209       {
210           remainder = (remainder << 1);
211       }
212     }
213   }
214   return (remainder);
215 }
216
217 void InitiateI2C()
218 {
219   Wire.beginTransmission(SlaveAddress);
220   Wire.write(lowByte(InitialAddr));
221   Wire.write(highByte(InitialAddr));
222 }
223
224 void PrintRegIndex(unsigned int partial)
225 {
226   unsigned int LineRegAdd = InitialAddr+partial;
227   Serial.println();
228   if (LineRegAdd < 0x1000) Serial.print("0");
229   if (LineRegAdd < 0x100) Serial.print("0");
230   if (LineRegAdd < 0x10) Serial.print("0");
231   Serial.print(LineRegAdd, HEX);
232   Serial.print(": ");
233 }
234
235 void TxCicle()
236 {
237   BufferLength = Wire.getTxBufferLength();
238   CRCtx = ComputeCRC(0x02);
239   Serial.print("TASK - I2C TX: ");
240   Serial.print(BufferLength);
241   Serial.print(" bytes - 1st Addr: 0x");
242   if (InitialAddr < 0x1000) Serial.print("0");
```

```
243   if (InitialAddr < 0x100) Serial.print("0");
244   if (InitialAddr < 0x10) Serial.print("0");
245   Serial.print(InitialAddr, HEX);
246   Serial.print(" CRC: 0x");
247   if (CRCtx < 0x10) Serial.print("0");
248   Serial.println(CRCtx, HEX);
249   Wire.endTransmission();
250 }
251
252 void I2Cscan()
253 {
254   byte I2CAddressScan = 0;
255   Serial.println("INFO - Slave scan start...");
256   while(I2CAddressScan < 127)
257   {
258     I2CAddressScan++;    // 0x00 are a "General Call" address, so we
skip it
259     Wire.beginTransmission(I2CAddressScan);
260     if (Wire.endTransmission() == 0)
261     {
262       SlaveAddress = I2CAddressScan;
263       Serial.print("INFO - Found Slave: 0x");
264       Serial.println(I2CAddressScan, HEX);
265     }
266   }
267   Serial.println("INFO - Slave scan end !");
268
269   if (SlaveAddress == 0)
270   {
271     Serial.println("ERRR - No device found! Please check your hard-
ware");
272     Serial.println("INFO - Press 'S' to try again");
273   }
274   else
275   {
276     Serial.print("INFO - Slave Address set to: 0x");
277     Serial.println(SlaveAddress, HEX);
278   }
279 }
280
281 void AbortInfo()
282 {
283   Serial.println("");
284   Serial.println("INFO - Current operation aborted");
285 }
286
287 byte convert(byte input)
288 {
289   illegalInput=false;
290   byte output;
291   switch (input)
292   {
293   case 0x30:
294     output=0x0;
295     break;
296   case 0x31:
297     output=0x1;
298     break;
299   case 0x32:
300     output=0x2;
301     break;
302   case 0x33:
303     output=0x3;
```

```
304     break;
305   case 0x34:
306     output=0x4;
307     break;
308   case 0x35:
309     output=0x5;
310     break;
311   case 0x36:
312     output=0x6;
313     break;
314   case 0x37:
315     output=0x7;
316     break;
317   case 0x38:
318     output=0x8;
319     break;
320   case 0x39:
321     output=0x9;
322     break;
323   case 0x41:
324     output=0xA;
325     break;
326   case 0x42:
327     output=0xB;
328     break;
329   case 0x43:
330     output=0xC;
331     break;
332   case 0x44:
333     output=0xD;
334     break;
335   case 0x45:
336     output=0xE;
337     break;
338   case 0x46:
339     output=0xF;
340     break;
341   case 0x61:
342     output=0xA;
343     break;
344   case 0x62:
345     output=0xB;
346     break;
347   case 0x63:
348     output=0xC;
349     break;
350   case 0x64:
351     output=0xD;
352     break;
353   case 0x65:
354     output=0xE;
355     break;
356   case 0x66:
357     output=0xF;
358     break;
359   default:
360     illegalInput=true;
361     break;
362   }
363   return output;
364 }
365
366 void ReadI2C(boolean Print)
```

```
367 {
368   InitiateI2C();
369   Wire.endTransmission();
370   Wire.flush();
371   Wire.requestFrom(SlaveAddress, NumberAddr+1);
372   BufferLength = Wire.getRxBufferLength();
373   CRCrx = ComputeCRC(0x00);
374   if(Print) {
375     byte RxSeq = 0x00;
376     while (Wire.available()) {
377       byte received=Wire.read();
378       if ((RxSeq&0x0F)==0x00) PrintRegIndex(RxSeq);
379       if (received < 0x10) Serial.print("0");
380       Serial.print(received, HEX);
381       Serial.print(" ");
382       RxSeq++;
383     }
384     Serial.println();
385     Serial.println();
386     Serial.print("TASK - I2C RX: ");
387     Serial.print(BufferLength);
388     Serial.print(" reg(s) - 1st: 0x");
389     if (InitialAddr < 0x1000) Serial.print("0");
390     if (InitialAddr < 0x100) Serial.print("0");
391     if (InitialAddr < 0x10) Serial.print("0");
392     Serial.print(InitialAddr, HEX);
393     Serial.print(" CRC: 0x");
394     if (CRCrx < 0x10) Serial.print("0");
395     Serial.println(CRCrx, HEX);
396   }
397 }
398
399 void CommandList()
400 {
401   Serial.println("INFO - Command list:");
402   Serial.println("       'S' -> Do a slave scan");
403   Serial.println("       'A' -> Change slave address (7-bit)");
404   Serial.println("       'W' -> Write data on Slave");
405   Serial.println("       'R' -> Read up to 256 bytes from Slave");
406   Serial.println("       ESC -> Abort current operation");
407 }
```

Apêndice F

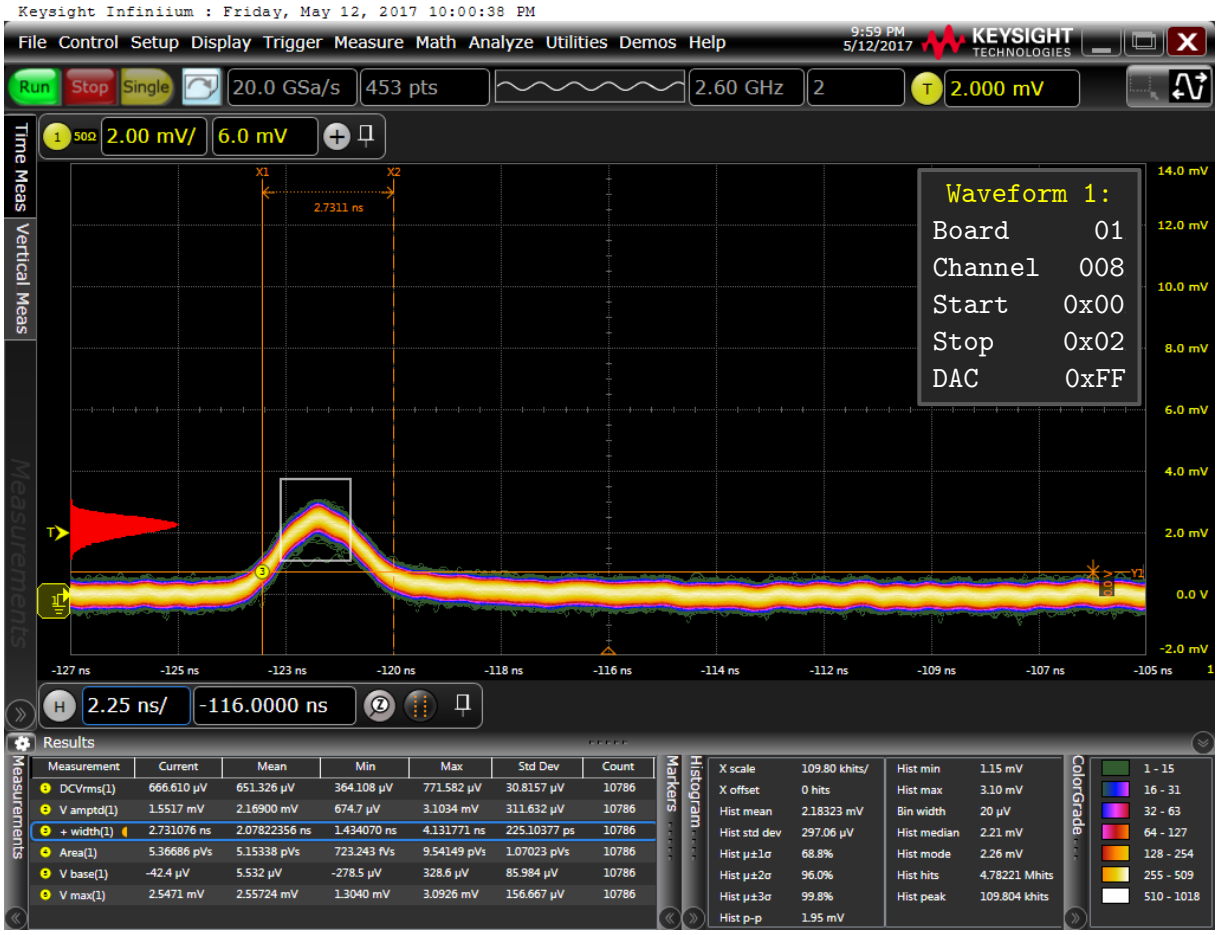
Formas de Onda na Saída do Injetor

Índice de Formas de Onda deste Apêndice:

Waveform 1: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0xFF 230
Waveform 2: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0xEE 230
Waveform 3: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0xDD 231
Waveform 4: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0xCC 231
Waveform 5: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0xBB 232
Waveform 6: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0xAA 232
Waveform 7: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x99 233
Waveform 8: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x88 233
Waveform 9: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x77 234
Waveform 10: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x66 234
Waveform 11: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x55 235
Waveform 12: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x44 235
Waveform 13: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x33 236
Waveform 14: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x22 236
Waveform 15: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x11 237
Waveform 16: Board 01; Channel 008; Start 0x00; Stop 0x02; DAC 0x00 237
Waveform 17: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0xFF 238
Waveform 18: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0xEE 238
Waveform 19: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0xDD 239
Waveform 20: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0xCC 239
Waveform 21: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0xBB 240

Waveform 22: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0xAA 240
Waveform 23: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x99 241
Waveform 24: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x88 241
Waveform 25: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x77 242
Waveform 26: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x66 242
Waveform 27: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x55 243
Waveform 28: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x44 243
Waveform 29: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x33 244
Waveform 30: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x22 244
Waveform 31: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x11 245
Waveform 32: Board 01; Channel 008; Start 0x00; Stop 0x03; DAC 0x00 245
Waveform 33: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0xFF 246
Waveform 34: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0xEE..... 246
Waveform 35: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0xDD 247
Waveform 36: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0xCC 247
Waveform 37: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0xBB 248
Waveform 38: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0xAA 248
Waveform 39: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x99 249
Waveform 40: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x88 249
Waveform 41: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x77 250
Waveform 42: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x66 250
Waveform 43: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x55 251
Waveform 44: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x44 251
Waveform 45: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x33 252
Waveform 46: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x22 252
Waveform 47: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x11 253
Waveform 48: Board 01; Channel 008; Start 0x00; Stop 0x04; DAC 0x00 253
Waveform 49: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0xFF 254

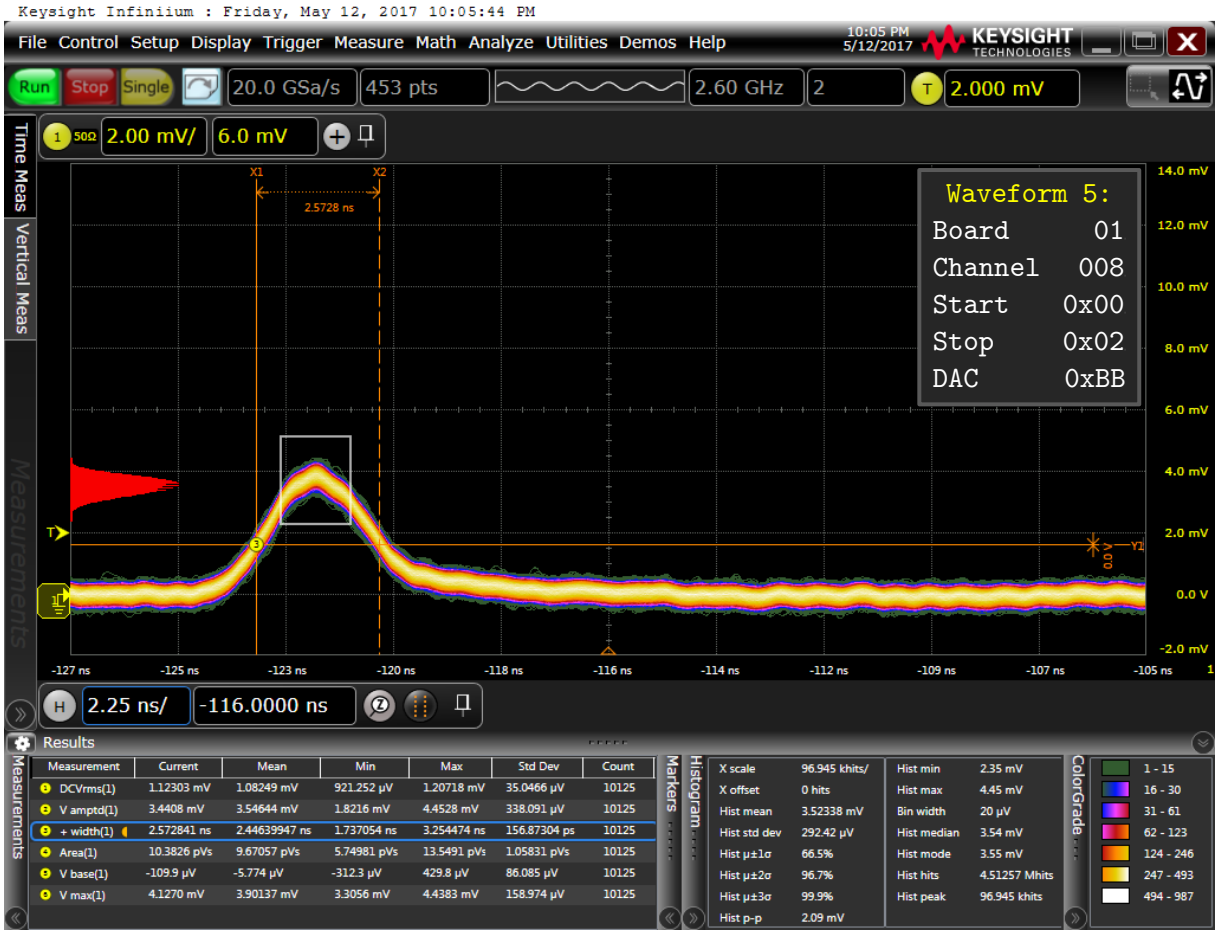
Waveform 50: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0xEE 254
Waveform 51: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0xDD..... 255
Waveform 52: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0xCC..... 255
Waveform 53: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0xBB..... 256
Waveform 54: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0xAA..... 256
Waveform 55: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x99 257
Waveform 56: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x88 257
Waveform 57: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x77 258
Waveform 58: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x66 258
Waveform 59: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x55 259
Waveform 60: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x44 259
Waveform 61: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x33..... 260
Waveform 62: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x22 260
Waveform 63: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x11 261
Waveform 64: Board 01; Channel 008; Start 0x00; Stop 0x05; DAC 0x00 261

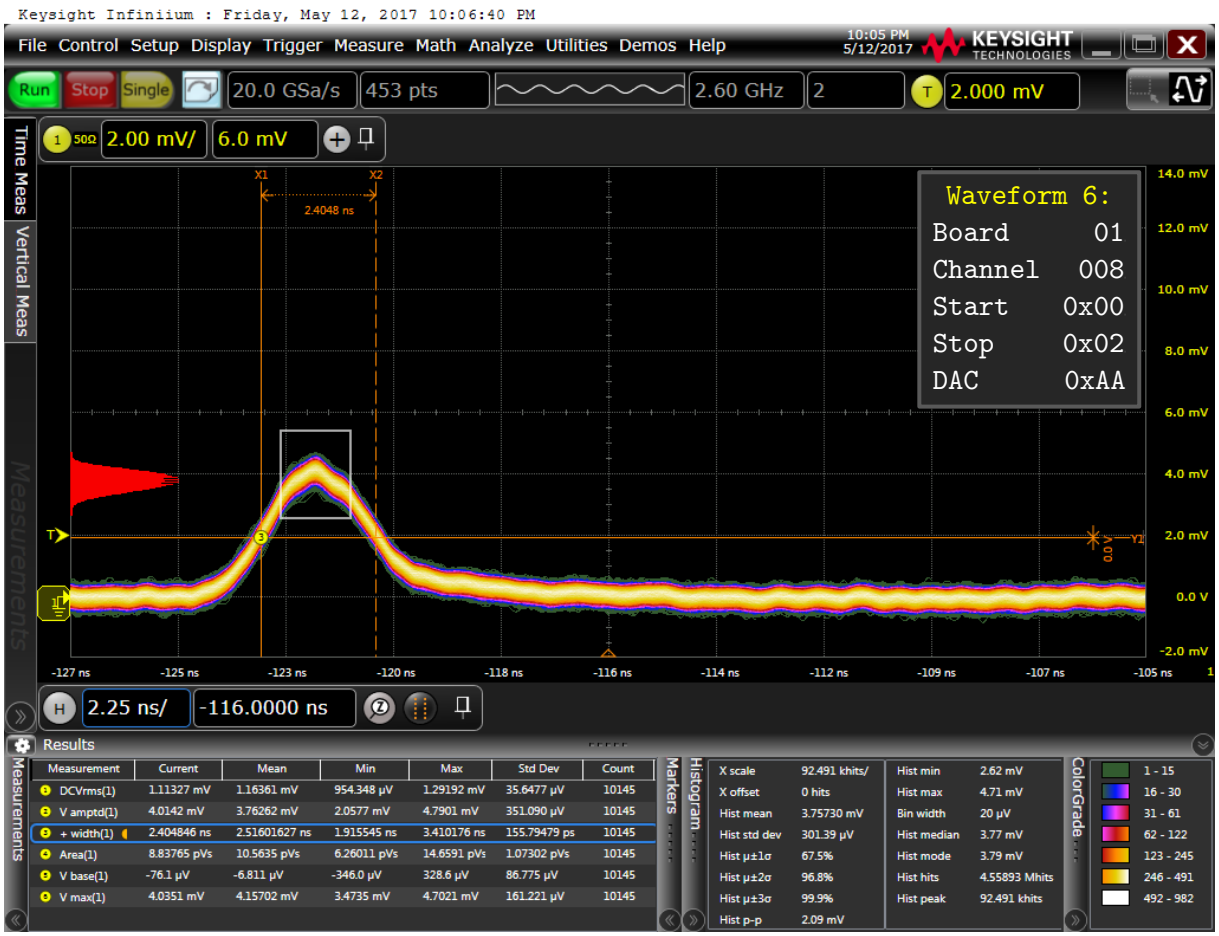

Keysight Infiniium : Friday, May 12, 2017 10:00:38 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
9:59 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.7311 ns
Waveform 1:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0xFF
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 666.610 µV 651.326 µV 364.108 µV 771.582 µV 30.8157 µV 10786
V amptd(1) 1.5517 mV 2.16900 mV 674.7 µV 3.1034 mV 311.632 µV 10786
+ width(1) 2.731076 ns 2.07822356 ns 1.434070 ns 4.131771 ns 225.10377 ps 10786
Area(1) 5.36686 pVs 5.15338 pVs 723.243 fVs 9.54149 pVs 1.07023 pVs 10786
V base(1) -42.4 µV 5.532 µV -278.5 µV 328.6 µV 85.984 µV 10786
V max(1) 2.5471 mV 2.55724 mV 1.3040 mV 3.0926 mV 156.667 µV 10786
X scale 109.80 khits/
X offset 0 hits
Hist mean 2.18323 mV
Hist std dev 297.06 µV
Hist µ±1σ 68.8%
Hist µ±2σ 96.0%
Hist µ±3σ 99.8%
Hist p-p 1.95 mV
Hist min 1.15 mV
Hist max 3.10 mV
Bin width 20 µV
Hist median 2.21 mV
Hist mode 2.26 mV
Hist hits 4.78221 Mhits
Hist peak 109.804 khits
1 - 15
16 - 31
32 - 63
64 - 127
128 - 254
255 - 509
510 - 1018

Keysight Infiniium : Friday, May 12, 2017 10:01:42 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:00 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.3527 ns
Waveform 2:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0xEE
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 706.482 µV 707.277 µV 568.559 µV 828.715 µV 31.5833 µV 10138
V amptd(1) 1.8216 mV 2.35842 mV 1.1807 mV 3.2721 mV 311.684 µV 10138
+ width(1) 2.352670 ns 2.12601271 ns 1.501576 ns 4.445417 ns 208.07215 ps 10138
Area(1) 5.44080 pVs 5.70117 pVs 1.66386 pVs 9.92359 pVs 1.04409 pVs 10138
V base(1) 25.1 µV 2.913 µV -346.0 µV 328.6 µV 85.410 µV 10138
V max(1) 2.7616 mV 2.74437 mV 2.1226 mV 3.2935 mV 159.407 µV 10138
X scale 101.37 khits/
X offset 0 hits
Hist mean 2.37917 mV
Hist std dev 281.91 µV
Hist µ±1σ 67.8%
Hist µ±2σ 96.3%
Hist µ±3σ 99.9%
Hist p-p 2.16 mV
Hist min 1.15 mV
Hist max 3.30 mV
Bin width 20 µV
Hist median 2.38 mV
Hist mode 2.46 mV
Hist hits 4.45813 Mhits
Hist peak 101.375 khits
1 - 15
16 - 30
31 - 61
62 - 122
123 - 244
245 - 489
490 - 979

Keysight Infiniium : Friday, May 12, 2017 10:02:46 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:02 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
X1 X2
1.9155 ns
Waveform 3:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0xDD
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 789.791 µV 801.565 µV 614.715 µV 920.872 µV 32.7893 µV 10115
V amptd(1) 3.2046 mV 2.68855 mV 1.4843 mV 3.7106 mV 322.250 µV 10115
+ width(1) 1.915502 ns 2.20434195 ns 1.529609 ns 3.489205 ns 194.91780 ps 10115
Area(1) 6.66406 pVs 6.57593 pVs 2.68051 pVs 10.6649 pVs 1.07296 pVs 10115
V base(1) -42.4 µV -4.874 µV -346.0 µV 328.6 µV 87.066 µV 10115
V max(1) 3.1925 mV 3.04929 mV 2.3484 mV 3.6434 mV 156.255 µV 10115
Histogram
X scale 99.566 khits/
X offset 0 hits
Hist mean 2.68410 mV
Hist std dev 282.45 µV
Hist µ±1σ 67.2%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 1.92 mV
Hist min 1.73 mV
Hist max 3.65 mV
Bin width 20 µV
Hist median 2.70 mV
Hist mode 2.76 mV
Hist hits 4.50831 Mhits
Hist peak 99.566 khits
ColorGrade
1 - 15
16 - 30
31 - 60
61 - 121
122 - 243
244 - 486
487 - 972

Keysight Infiniium : Friday, May 12, 2017 10:03:59 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:03 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
X1 X2
2.4285 ns
Waveform 4:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0xCC
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 899.843 µV 979.308 µV 837.180 µV 1.08842 mV 32.3295 µV 10185
V amptd(1) 3.0022 mV 3.23794 mV 1.8216 mV 4.1154 mV 333.074 µV 10185
+ width(1) 2.428463 ns 2.36579612 ns 1.726683 ns 3.297378 ns 169.57189 ps 10185
Area(1) 7.26678 pVs 8.44674 pVs 4.02201 pVs 12.9334 pVs 1.05335 pVs 10185
V base(1) -8.7 µV -7.941 µV -312.3 µV 294.9 µV 86.417 µV 10185
V max(1) 3.2313 mV 3.59210 mV 3.0502 mV 4.0944 mV 152.993 µV 10185
Histogram
X scale 101.01 khits/
X offset 0 hits
Hist mean 3.21999 mV
Hist std dev 287.63 µV
Hist µ±1σ 66.1%
Hist µ±2σ 96.7%
Hist µ±3σ 99.9%
Hist p-p 2.25 mV
Hist min 1.85 mV
Hist max 4.10 mV
Bin width 20 µV
Hist median 3.24 mV
Hist mode 3.29 mV
Hist hits 4.53555 Mhits
Hist peak 101.017 khits
ColorGrade
1 - 15
16 - 31
32 - 63
64 - 126
127 - 252
253 - 504
505 - 1009


Keysight Infiniium : Friday, May 12, 2017 10:05:44 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:05 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.5728 ns
Waveform 5:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0xBB
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.12303 mV 1.08249 mV 921.252 µV 1.20718 mV 35.0466 µV 10125
V amptd(1) 3.4408 mV 3.54644 mV 1.8216 mV 4.4528 mV 338.091 µV 10125
+ width(1) 2.572841 ns 2.44639947 ns 1.737054 ns 3.254474 ns 156.87304 ps 10125
Area(1) 10.3826 pVs 9.67057 pVs 5.74981 pVs 13.5491 pVs 1.05831 pVs 10125
V base(1) -109.9 µV -5.774 mV -312.3 µV 429.8 µV 86.085 µV 10125
V max(1) 4.1270 mV 3.90137 mV 3.3056 mV 4.4383 mV 158.974 µV 10125
X scale 96.945 khits/
X offset 0 hits
Hist mean 3.52338 mV
Hist std dev 292.42 µV
Hist µ±1σ 66.5%
Hist µ±2σ 96.7%
Hist µ±3σ 99.9%
Hist p-p 2.09 mV
Hist min 2.35 mV
Hist max 4.45 mV
Bin width 20 µV
Hist median 3.54 mV
Hist mode 3.55 mV
Hist hits 4.51257 Mhits
Hist peak 96.945 khits
1 - 15
16 - 30
31 - 61
62 - 123
124 - 246
247 - 493
494 - 987


Keysight Infiniium : Friday, May 12, 2017 10:06:40 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:05 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.4048 ns
Waveform 6:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0xAA
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.11327 mV 1.16361 mV 954.348 µV 1.29192 mV 35.6477 µV 10145
V amptd(1) 4.0142 mV 3.76262 mV 2.0577 mV 4.7901 mV 351.090 µV 10145
+ width(1) 2.404846 ns 2.51601627 ns 1.915545 ns 3.410176 ns 155.79479 ps 10145
Area(1) 8.83765 pVs 10.5635 pVs 6.26011 pVs 14.6591 pVs 1.07302 pVs 10145
V base(1) -76.1 µV -6.811 µV -346.0 µV 328.6 µV 86.775 µV 10145
V max(1) 4.0351 mV 4.15702 mV 3.4735 mV 4.7021 mV 161.221 µV 10145
X scale 92.491 khits/
X offset 0 hits
Hist mean 3.75730 mV
Hist std dev 301.39 µV
Hist µ±1σ 67.5%
Hist µ±2σ 96.8%
Hist µ±3σ 99.9%
Hist p-p 2.09 mV
Hist min 2.62 mV
Hist max 4.71 mV
Bin width 20 µV
Hist median 3.77 mV
Hist mode 3.79 mV
Hist hits 4.55893 Mhits
Hist peak 92.491 khits
1 - 15
16 - 30
31 - 61
62 - 122
123 - 245
246 - 491
492 - 982

Keysight Infiniium : Friday, May 12, 2017 10:07:48 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:07 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2.000 mV
2.00 mV/
6.0 mV
2.6062 ns
Waveform 7:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x99
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.19914 mV 1.24411 mV 1.05863 mV 1.39448 mV 37.7922 µV 10190
V amptd(1) 4.1154 mV 4.02270 mV 2.2938 mV 4.9587 mV 357.975 µV 10190
+ width(1) 2.606234 ns 2.56133722 ns 2.055082 ns 3.407522 ns 153.17311 ps 10190
Area(1) 10.2806 pVs 11.4681 pVs 7.36790 pVs 15.4294 pVs 1.09264 pVs 10190
V base(1) -42.4 µV -6.537 µV -346.0 µV 362.4 µV 88.214 µV 10190
V max(1) 4.1060 mV 4.37956 mV 3.6964 mV 5.0054 mV 159.889 µV 10190
Histogram
X scale 91.898 khits/
X offset 0 hits
Hist mean 3.93446 mV
Hist std dev 350.11 µV
Hist µ±1σ 68.0%
Hist µ±2σ 96.3%
Hist µ±3σ 99.8%
Hist p-p 2.39 mV
Hist min 2.62 mV
Hist max 5.01 mV
Bin width 20 µV
Hist median 3.98 mV
Hist mode 4.09 mV
Hist hits 4.88144 Mhits
Hist peak 91.898 khits
ColorGrade
1 - 15
16 - 30
31 - 61
62 - 122
123 - 244
245 - 489
490 - 978

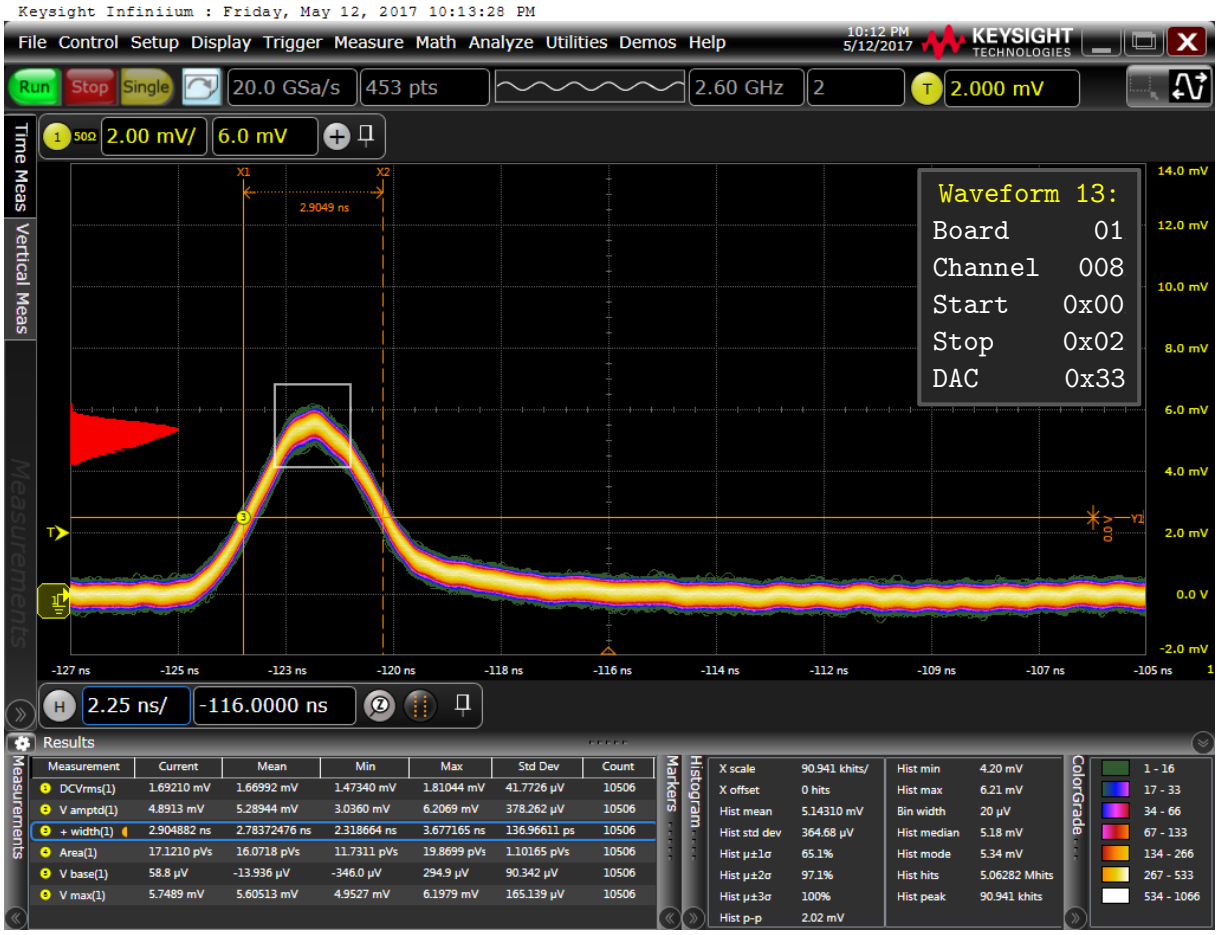
Keysight Infiniium : Friday, May 12, 2017 10:08:46 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:08 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2.000 mV
2.00 mV/
6.0 mV
2.5691 ns
Waveform 8:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x88
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.30520 mV 1.32189 mV 1.15897 mV 1.47925 mV 39.2506 µV 10157
V amptd(1) 3.7781 mV 4.25370 mV 2.4288 mV 5.1274 mV 362.721 µV 10157
+ width(1) 2.569087 ns 2.61180911 ns 2.166218 ns 3.501394 ns 146.19323 ps 10157
Area(1) 11.4372 pVs 12.2461 pVs 8.15315 pVs 16.8315 pVs 1.10491 pVs 10157
V base(1) -8.7 µV -13.180 µV -312.3 µV 328.6 µV 88.344 µV 10157
V max(1) 4.8826 mV 4.60648 mV 3.9729 mV 5.1809 mV 166.382 µV 10157
Histogram
X scale 91.610 khits/
X offset 0 hits
Hist mean 4.15995 mV
Hist std dev 351.86 µV
Hist µ±1σ 67.7%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 2.19 mV
Hist min 3.01 mV
Hist max 5.20 mV
Bin width 20 µV
Hist median 4.20 mV
Hist mode 4.27 mV
Hist hits 4.92177 Mhits
Hist peak 91.610 khits
ColorGrade
1 - 15
16 - 31
32 - 62
63 - 125
126 - 251
252 - 502
503 - 1005

Keysight Infiniium : Friday, May 12, 2017 10:09:40 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:08 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.7404 ns
Waveform 9:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x77
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.30860 mV 1.38401 mV 1.17694 mV 1.52668 mV 37.7823 µV 10221
V amptd(1) 3.9130 mV 4.43431 mV 2.4962 mV 5.2961 mV 376.105 µV 10221
+ width(1) 2.740375 ns 2.64060262 ns 2.179021 ns 3.628984 ns 148.02606 ps 10221
Area(1) 12.4101 pVs 12.9038 pVs 8.62650 pVs 16.8605 pVs 1.07882 pVs 10221
V base(1) -42.4 µV -7.557 µV -379.7 µV 328.6 µV 91.533 µV 10221
V max(1) 4.4963 mV 4.79003 mV 4.1537 mV 5.4012 mV 160.765 µV 10221
X scale 90.169 khits/
X offset 0 hits
Hist mean 4.33667 mV
Hist std dev 357.60 µV
Hist µ±1σ 66.2%
Hist µ±2σ 96.4%
Hist µ±3σ 100%
Hist p-p 2.08 mV
Hist min 3.23 mV
Hist max 5.30 mV
Bin width 20 µV
Hist median 4.37 mV
Hist mode 4.51 mV
Hist hits 4.89995 Mhits
Hist peak 90.169 khits
1 - 15
16 - 30
31 - 61
62 - 122
123 - 245
246 - 490
491 - 980

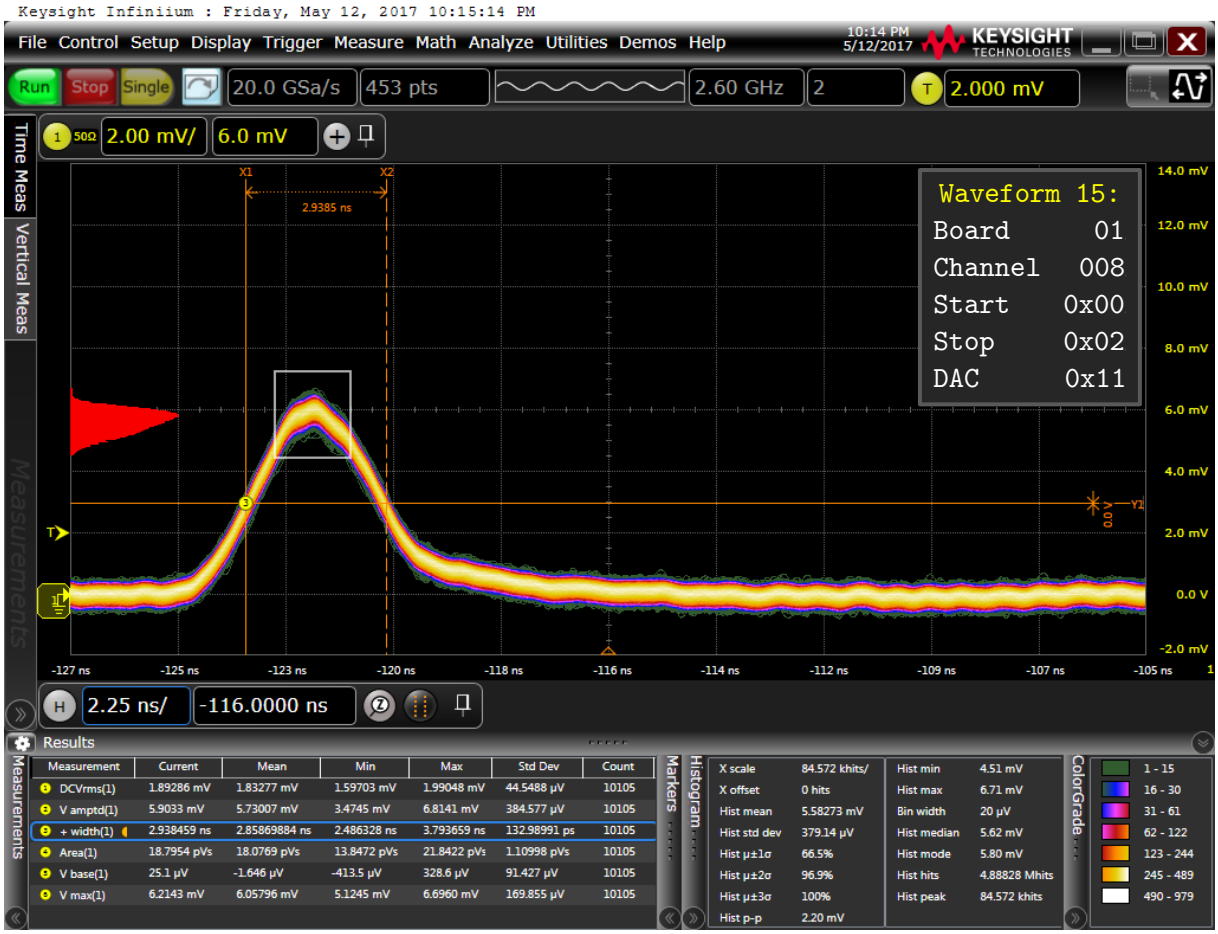
Keysight Infiniium : Friday, May 12, 2017 10:10:38 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:09 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.6700 ns
Waveform 10:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x66
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.43032 mV 1.45437 mV 1.24685 mV 1.59359 mV 39.3263 µV 10136
V amptd(1) 4.7226 mV 4.65187 mV 2.6986 mV 5.4985 mV 376.679 µV 10136
+ width(1) 2.670032 ns 2.67927212 ns 2.179473 ns 3.549138 ns 144.04320 ps 10136
Area(1) 14.8342 pVs 13.6948 pVs 8.65213 pVs 17.9296 pVs 1.08117 pVs 10136
V base(1) -8.7 µV -12.596 µV -346.0 µV 328.6 µV 89.887 µV 10136
V max(1) 4.7366 mV 4.99222 mV 4.2145 mV 5.5725 mV 165.994 µV 10136
X scale 89.428 khits/
X offset 0 hits
Hist mean 4.53629 mV
Hist std dev 360.11 µV
Hist µ±1σ 66.9%
Hist µ±2σ 96.5%
Hist µ±3σ 100%
Hist p-p 2.13 mV
Hist min 3.46 mV
Hist max 5.59 mV
Bin width 20 µV
Hist median 4.57 mV
Hist mode 4.74 mV
Hist hits 4.89964 Mhits
Hist peak 89.428 khits
1 - 15
16 - 30
31 - 60
61 - 121
122 - 243
244 - 487
488 - 975

Keysight Infiniium : Friday, May 12, 2017 10:11:32 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:10 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2.000 mV
2.00 mV/
6.0 mV
2.7162 ns
Waveform 11:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x55
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.54572 mV 1.53421 mV 1.34582 mV 1.67284 mV 41.0758 µV 10361
V amptd(1) 4.8913 mV 4.89494 mV 2.7998 mV 5.7346 mV 368.259 µV 10361
+ width(1) 2.716217 ns 2.72061195 ns 2.295194 ns 3.594261 ns 136.94112 ps 10361
Area(1) 15.7675 pVs 14.5974 pVs 9.72883 pVs 18.6823 pVs 1.09702 pVs 10361
V base(1) 126.2 µV -14.449 µV -379.7 µV 328.6 µV 89.312 µV 10361
V max(1) 5.1886 mV 5.21599 mV 4.6457 mV 5.8427 mV 167.132 µV 10361
X scale 91.262 khits/
X offset 0 hits
Hist mean 4.75787 mV
Hist std dev 365.36 µV
Hist µ±1σ 67.4%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 2.39 mV
Hist min 3.46 mV
Hist max 5.85 mV
Bin width 20 µV
Hist median 4.79 mV
Hist mode 4.93 mV
Hist hits 5.02777 Mhits
Hist peak 91.262 khits
1 - 15
16 - 31
32 - 62
63 - 125
126 - 250
251 - 500
501 - 1001

Keysight Infiniium : Friday, May 12, 2017 10:12:29 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:11 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2.000 mV
2.00 mV/
6.0 mV
2.8498 ns
Waveform 12:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x44
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.59589 mV 1.59987 mV 1.41786 mV 1.74971 mV 40.7806 µV 10798
V amptd(1) 5.1611 mV 5.08788 mV 2.9685 mV 5.9370 mV 369.716 µV 10798
+ width(1) 2.849774 ns 2.74854037 ns 2.349435 ns 3.610961 ns 135.72543 ps 10798
Area(1) 16.0728 pVs 15.3401 pVs 10.8918 pVs 19.1883 pVs 1.08407 pVs 10798
V base(1) -76.1 µV -12.282 µV -447.2 µV 294.9 µV 89.547 µV 10798
V max(1) 5.2494 mV 5.40343 mV 4.6673 mV 5.9700 mV 166.284 µV 10798
X scale 92.993 khits/
X offset 0 hits
Hist mean 4.94194 mV
Hist std dev 366.08 µV
Hist µ±1σ 66.7%
Hist µ±2σ 96.9%
Hist µ±3σ 100%
Hist p-p 2.13 mV
Hist min 3.85 mV
Hist max 5.98 mV
Bin width 20 µV
Hist median 4.98 mV
Hist mode 5.15 mV
Hist hits 5.24128 Mhits
Hist peak 92.993 khits
1 - 16
17 - 33
34 - 67
68 - 134
135 - 268
269 - 537
538 - 1075


Keysight Infiniium : Friday, May 12, 2017 10:13:28 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:12 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
X1 X2
2.9049 ns
Waveform 13:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x33
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.69210 mV 1.66992 mV 1.47340 mV 1.81044 mV 41.7726 µV 10506
V amptd(1) 4.8913 mV 5.28944 mV 3.0360 mV 6.2069 mV 378.262 µV 10506
+ width(1) 2.904882 ns 2.78372476 ns 2.318664 ns 3.677165 ns 136.96611 ps 10506
Area(1) 17.1210 pVs 16.0718 pVs 11.7311 pVs 19.8699 pVs 1.10165 pVs 10506
V base(1) 58.8 µV -13.936 µV -346.0 µV 294.9 µV 90.342 µV 10506
V max(1) 5.7489 mV 5.60513 mV 4.9527 mV 6.1979 mV 165.139 µV 10506
X scale 90.941 khits/
X offset 0 hits
Hist mean 5.14310 mV
Hist std dev 364.68 µV
Hist µ±1σ 65.1%
Hist µ±2σ 97.1%
Hist µ±3σ 100%
Hist p-p 2.02 mV
Hist min 4.20 mV
Hist max 6.21 mV
Bin width 20 µV
Hist median 5.18 mV
Hist mode 5.34 mV
Hist hits 5.06282 Mhits
Hist peak 90.941 khits
1 - 16
17 - 33
34 - 66
67 - 133
134 - 266
267 - 533
534 - 1066

Keysight Infiniium : Friday, May 12, 2017 10:14:21 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:13 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
X1 X2
2.9686 ns
Waveform 14:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x22
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.77806 mV 1.75068 mV 1.54184 mV 1.89793 mV 42.4155 µV 10138
V amptd(1) 5.1949 mV 5.51139 mV 3.2721 mV 6.4767 mV 380.485 µV 10138
+ width(1) 2.968603 ns 2.82084365 ns 2.371560 ns 3.798973 ns 133.79454 ps 10138
Area(1) 17.0356 pVs 17.1558 pVs 12.6311 pVs 21.6438 pVs 1.10340 pVs 10138
V base(1) 58.8 µV -7.771 µV -346.0 µV 328.6 µV 91.248 µV 10138
V max(1) 5.9910 mV 5.82766 mV 5.1140 mV 6.3426 mV 166.883 µV 10138
X scale 87.156 khits/
X offset 0 hits
Hist mean 5.35701 mV
Hist std dev 375.15 µV
Hist µ±1σ 66.3%
Hist µ±2σ 96.8%
Hist µ±3σ 100%
Hist p-p 2.16 mV
Hist min 4.20 mV
Hist max 6.35 mV
Bin width 20 µV
Hist median 5.40 mV
Hist mode 5.57 mV
Hist hits 4.88752 Mhits
Hist peak 87.156 khits
1 - 15
16 - 31
32 - 62
63 - 125
126 - 250
251 - 500
501 - 1001


Keysight Infiniium : Friday, May 12, 2017 10:15:14 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:14 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.9385 ns
Waveform 15:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x11
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.89286 mV 1.83277 mV 1.59703 mV 1.99048 mV 44.5488 µV 10105
V amptd(1) 5.9033 mV 5.73007 mV 3.4745 mV 6.8141 mV 384.577 µV 10105
+ width(1) 2.938459 ns 2.85869884 ns 2.486328 ns 3.793659 ns 132.98991 ps 10105
Area(1) 18.7954 pVs 18.0769 pVs 13.8472 pVs 21.8422 pVs 1.10998 pVs 10105
V base(1) 25.1 µV -1.646 µV -413.5 µV 328.6 µV 91.427 µV 10105
V max(1) 6.2143 mV 6.05796 mV 5.1245 mV 6.6960 mV 169.855 µV 10105
X scale 84.572 khits/
X offset 0 hits
Hist mean 5.58273 mV
Hist std dev 379.14 µV
Hist µ±1σ 66.5%
Hist µ±2σ 96.9%
Hist µ±3σ 100%
Hist p-p 2.20 mV
Hist min 4.51 mV
Hist max 6.71 mV
Bin width 20 µV
Hist median 5.62 mV
Hist mode 5.80 mV
Hist hits 4.88828 Mhits
Hist peak 84.572 khits
1 - 15
16 - 30
31 - 61
62 - 122
123 - 244
245 - 489
490 - 979

Keysight Infiniium : Friday, May 12, 2017 10:16:01 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:15 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
2.8158 ns
Waveform 16:
Board 01
Channel 008
Start 0x00
Stop 0x02
DAC 0x00
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.90317 mV 1.91092 mV 1.71114 mV 2.09090 mV 45.5635 µV 10155
V amptd(1) 6.3755 mV 5.96751 mV 3.5420 mV 6.8478 mV 377.879 µV 10155
+ width(1) 2.815833 ns 2.89150123 ns 2.523790 ns 3.723362 ns 128.96516 ps 10155
Area(1) 17.4730 pVs 18.9488 pVs 14.7786 pVs 23.2502 pVs 1.14376 pVs 10155
V base(1) -8.7 µV -5.829 µV -312.3 µV 328.6 µV 91.062 µV 10155
V max(1) 6.3911 mV 6.27649 mV 5.5306 mV 6.8786 mV 168.650 µV 10155
X scale 82.895 khits/
X offset 0 hits
Hist mean 5.79434 mV
Hist std dev 386.01 µV
Hist µ±1σ 66.5%
Hist µ±2σ 97.2%
Hist µ±3σ 99.9%
Hist p-p 2.36 mV
Hist min 4.51 mV
Hist max 6.87 mV
Bin width 20 µV
Hist median 5.84 mV
Hist mode 6.02 mV
Hist hits 4.88735 Mhits
Hist peak 82.895 khits
1 - 15
16 - 30
31 - 61
62 - 123
124 - 246
247 - 492
493 - 985

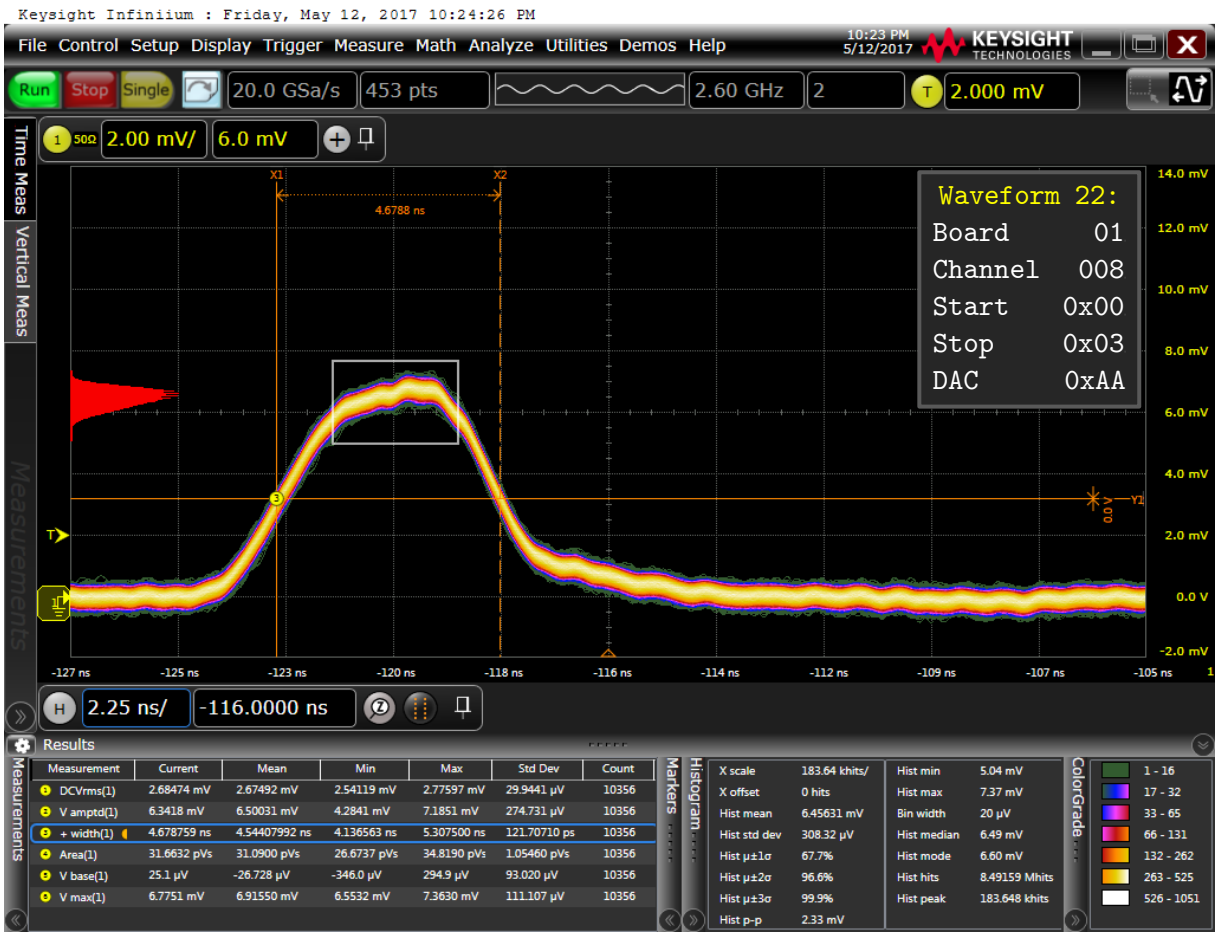
Keysight Infiniium : Friday, May 12, 2017 10:17:35 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:16 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.3055 ns
Waveform 17:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0xFF
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.73595 mV 1.76226 mV 1.65217 mV 1.92050 mV 28.7383 µV 10178
V amptd(1) 4.1154 mV 4.37921 mV 2.9685 mV 4.9925 mV 232.247 µV 10178
+ width(1) 4.305500 ns 4.16721031 ns 3.653162 ns 4.825223 ns 124.28191 ps 10178
Area(1) 18.9473 pVs 19.4606 pVs 15.3565 pVs 23.9056 pVs 1.05687 pVs 10178
V base(1) -42.4 µV -14.211 µV -346.0 µV 294.9 µV 90.037 µV 10178
V max(1) 4.7814 mV 4.77028 mV 4.4061 mV 5.1973 mV 104.763 µV 10178
X scale 213.35 khits/
X offset 0 hits
Hist mean 4.25818 mV
Hist std dev 366.29 µV
Hist µ±1σ 69.9%
Hist µ±2σ 95.4%
Hist µ±3σ 99.6%
Hist p-p 2.44 mV
Hist min 2.77 mV
Hist max 5.21 mV
Bin width 20 µV
Hist median 4.32 mV
Hist mode 4.46 mV
Hist hits 9.98571 Mhits
Hist peak 213.359 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 252
253 - 505
506 - 1010

Keysight Infiniium : Friday, May 12, 2017 10:19:27 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:18 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.1501 ns
Waveform 18:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0xEE
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 1.94082 mV 1.92437 mV 1.81727 mV 2.03170 mV 29.2578 µV 10160
V amptd(1) 4.6889 mV 4.78847 mV 3.1709 mV 5.4647 mV 240.020 µV 10160
+ width(1) 4.150096 ns 4.18978348 ns 3.680086 ns 5.003016 ns 121.27891 ps 10160
Area(1) 22.5863 pVs 21.5252 pVs 17.4083 pVs 25.1677 pVs 1.04750 pVs 10160
V base(1) 25.1 µV -10.741 µV -346.0 µV 328.6 µV 87.816 µV 10160
V max(1) 5.2906 mV 5.18210 mV 4.8394 mV 5.5913 mV 107.547 µV 10160
X scale 209.39 khits/
X offset 0 hits
Hist mean 4.65303 mV
Hist std dev 381.17 µV
Hist µ±1σ 68.3%
Hist µ±2σ 95.8%
Hist µ±3σ 100%
Hist p-p 2.09 mV
Hist min 3.51 mV
Hist max 5.60 mV
Bin width 20 µV
Hist median 4.73 mV
Hist mode 4.88 mV
Hist hits 9.89745 Mhits
Hist peak 209.396 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 252
253 - 504
505 - 1009

Keysight Infiniium : Friday, May 12, 2017 10:21:22 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:20 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.1557 ns
Waveform 19:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0xDD
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.10314 mV 2.10161 mV 1.97374 mV 2.23430 mV 28.8307 µV 10218
V amptd(1) 5.1949 mV 5.21771 mV 3.6432 mV 5.8358 mV 240.406 µV 10218
+ width(1) 4.155658 ns 4.26615672 ns 3.833784 ns 4.893845 ns 118.80078 ps 10218
Area(1) 23.2753 pVs 23.7084 pVs 19.2968 pVs 28.6006 pVs 1.04249 pVs 10218
V base(1) 92.5 µV -16.130 µV -379.7 µV 362.4 µV 91.462 µV 10218
V max(1) 5.6305 mV 5.60369 mV 5.2587 mV 6.0374 mV 107.716 µV 10218
X scale 202.03 khits/
X offset 0 hits
Hist mean 5.18640 mV
Hist std dev 280.55 µV
Hist µ±1σ 68.4%
Hist µ±2σ 96.2%
Hist µ±3σ 99.9%
Hist p-p 2.03 mV
Hist min 4.01 mV
Hist max 6.04 mV
Bin width 20 µV
Hist median 5.21 mV
Hist mode 5.30 mV
Hist hits 8.28471 Mhits
Hist peak 202.034 khits
1 - 16
17 - 32
33 - 64
65 - 128
129 - 256
257 - 512
513 - 1024

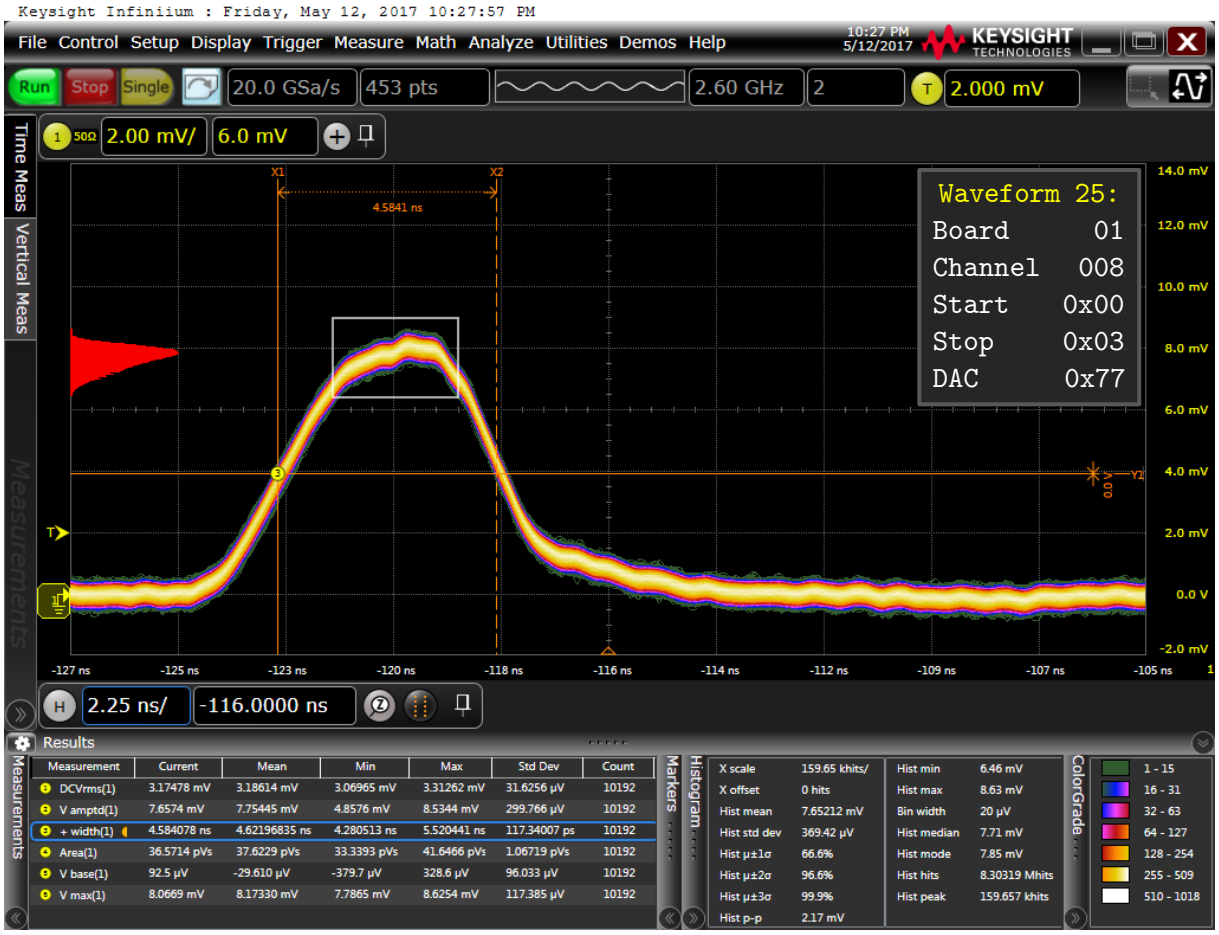
Keysight Infiniium : Friday, May 12, 2017 10:22:25 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:21 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.2145 ns
Waveform 20:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0xCC
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.27727 mV 2.32297 mV 2.20907 mV 2.43544 mV 29.5003 µV 10986
V amptd(1) 6.0045 mV 5.67006 mV 3.4745 mV 6.3418 mV 255.155 µV 10986
+ width(1) 4.214534 ns 4.44183041 ns 4.072870 ns 5.275066 ns 122.13469 ps 10986
Area(1) 25.5044 pVs 26.6721 pVs 22.3394 pVs 31.4988 pVs 1.04711 pVs 10986
V base(1) -42.4 µV -19.970 µV -346.0 µV 294.9 µV 92.354 µV 10986
V max(1) 5.9926 mV 6.06654 mV 5.7378 mV 6.4717 mV 107.883 µV 10986
X scale 212.87 khits/
X offset 0 hits
Hist mean 5.64792 mV
Hist std dev 278.33 µV
Hist µ±1σ 66.8%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 2.00 mV
Hist min 4.48 mV
Hist max 6.48 mV
Bin width 20 µV
Hist median 5.68 mV
Hist mode 5.76 mV
Hist hits 8.97435 Mhits
Hist peak 212.873 khits
1 - 16
17 - 33
34 - 67
68 - 134
135 - 269
270 - 538
539 - 1076

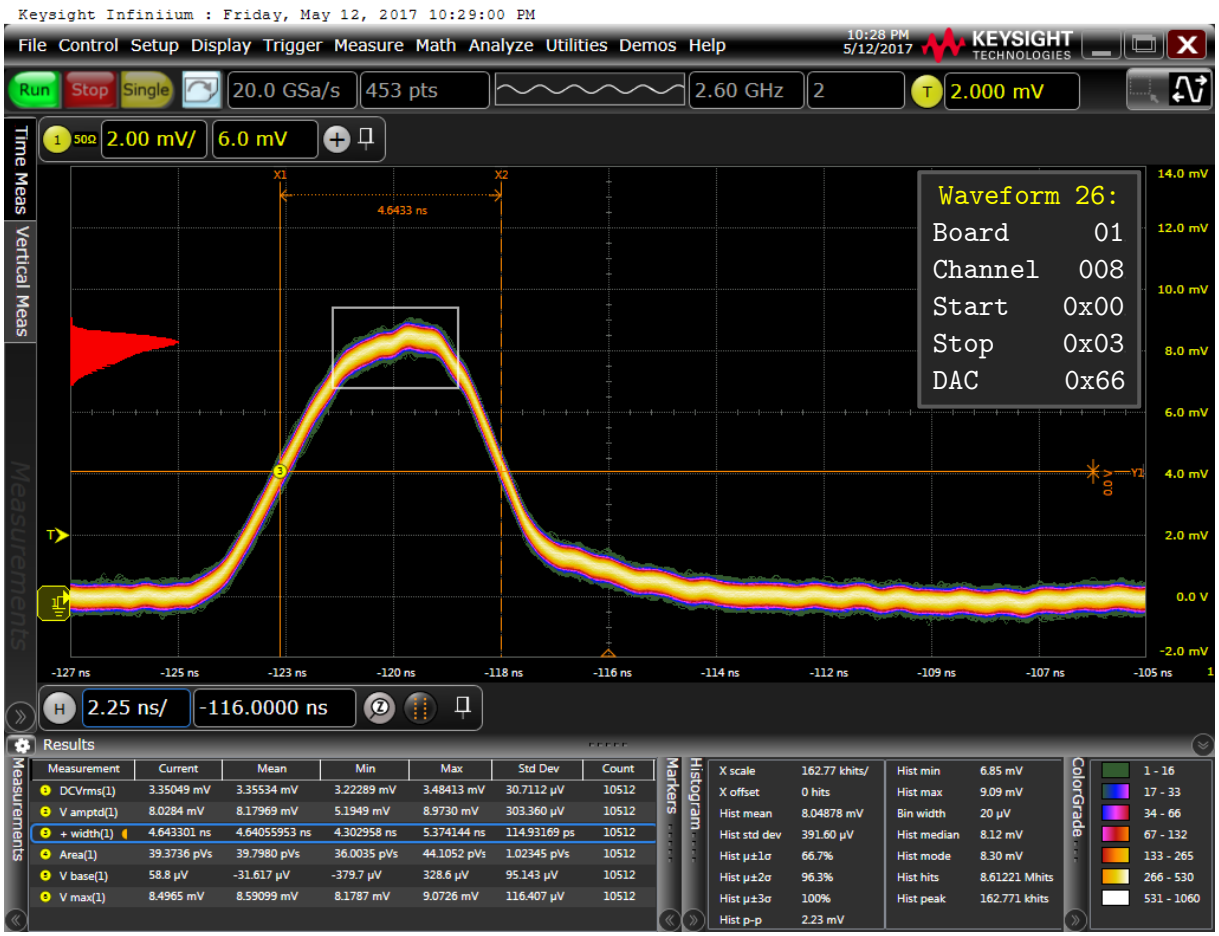
Keysight Infiniium : Friday, May 12, 2017 10:23:26 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:22 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.4582 ns
Waveform 21:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0xBB
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.46226 mV 2.49733 mV 2.38014 mV 2.60762 mV 29.6060 µV 10129
V amptd(1) 6.1394 mV 6.08645 mV 3.6769 mV 6.7466 mV 259.883 µV 10129
+ width(1) 4.458180 ns 4.50094580 ns 4.169974 ns 5.456594 ns 119.12733 ps 10129
Area(1) 27.5360 pVs 28.7686 pVs 24.1291 pVs 32.1834 pVs 1.04578 pVs 10129
V base(1) -177.3 µV -29.188 µV -379.7 µV 396.1 µV 93.117 µV 10129
V max(1) 6.4941 mV 6.48092 mV 6.0872 mV 6.9701 mV 110.235 µV 10129
X scale 187.59 khits/
X offset 0 hits
Hist mean 6.04628 mV
Hist std dev 290.43 µV
Hist µ±1σ 68.4%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 2.13 mV
Hist min 4.85 mV
Hist max 6.98 mV
Bin width 20 µV
Hist median 6.07 mV
Hist mode 6.15 mV
Hist hits 8.15745 Mhits
Hist peak 187.598 khits
1 - 15
16 - 30
31 - 61
62 - 123
124 - 246
247 - 493
494 - 986


Keysight Infiniium : Friday, May 12, 2017 10:24:26 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:23 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6788 ns
Waveform 22:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0xAA
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.68474 mV 2.67492 mV 2.54119 mV 2.77597 mV 29.9441 µV 10356
V amptd(1) 6.3418 mV 6.50031 mV 4.2841 mV 7.1851 mV 274.731 µV 10356
+ width(1) 4.678759 ns 4.54407992 ns 4.136563 ns 5.307500 ns 121.70710 ps 10356
Area(1) 31.6632 pVs 31.0900 pVs 26.6737 pVs 34.8190 pVs 1.05460 pVs 10356
V base(1) 25.1 µV -26.728 µV -346.0 µV 294.9 µV 93.020 µV 10356
V max(1) 6.7751 mV 6.91550 mV 6.5532 mV 7.3630 mV 111.107 µV 10356
X scale 183.64 khits/
X offset 0 hits
Hist mean 6.45631 mV
Hist std dev 308.32 µV
Hist µ±1σ 67.7%
Hist µ±2σ 96.6%
Hist µ±3σ 99.9%
Hist p-p 2.33 mV
Hist min 5.04 mV
Hist max 7.37 mV
Bin width 20 µV
Hist median 6.49 mV
Hist mode 6.60 mV
Hist hits 8.49159 Mhits
Hist peak 183.648 khits
1 - 16
17 - 32
33 - 65
66 - 131
132 - 262
263 - 525
526 - 1051

Keysight Infiniium : Friday, May 12, 2017 10:25:29 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:24 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6890 ns
Waveform 23:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x99
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.86631 mV 2.84830 mV 2.73370 mV 2.97033 mV 30.4494 µV 10103
V amptd(1) 6.8478 mV 6.92135 mV 4.8238 mV 7.6237 mV 281.876 µV 10103
+ width(1) 4.688968 ns 4.57275224 ns 4.191887 ns 5.499962 ns 119.77923 ps 10103
Area(1) 33.9763 pVs 33.3705 pVs 29.9034 pVs 37.0861 pVs 1.03960 pVs 10103
V base(1) -8.7 µV -25.254 µV -379.7 µV 362.4 µV 92.198 µV 10103
V max(1) 7.4442 mV 7.33737 mV 6.9198 mV 7.8187 mV 113.496 µV 10103
X scale 167.60 khits/
X offset 0 hits
Hist mean 6.86162 mV
Hist std dev 326.66 µV
Hist µ±1σ 67.3%
Hist µ±2σ 96.7%
Hist µ±3σ 99.9%
Hist p-p 2.17 mV
Hist min 5.65 mV
Hist max 7.82 mV
Bin width 20 µV
Hist median 6.90 mV
Hist mode 6.98 mV
Hist hits 8.12283 Mhits
Hist peak 167.600 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 253
254 - 507
508 - 1014

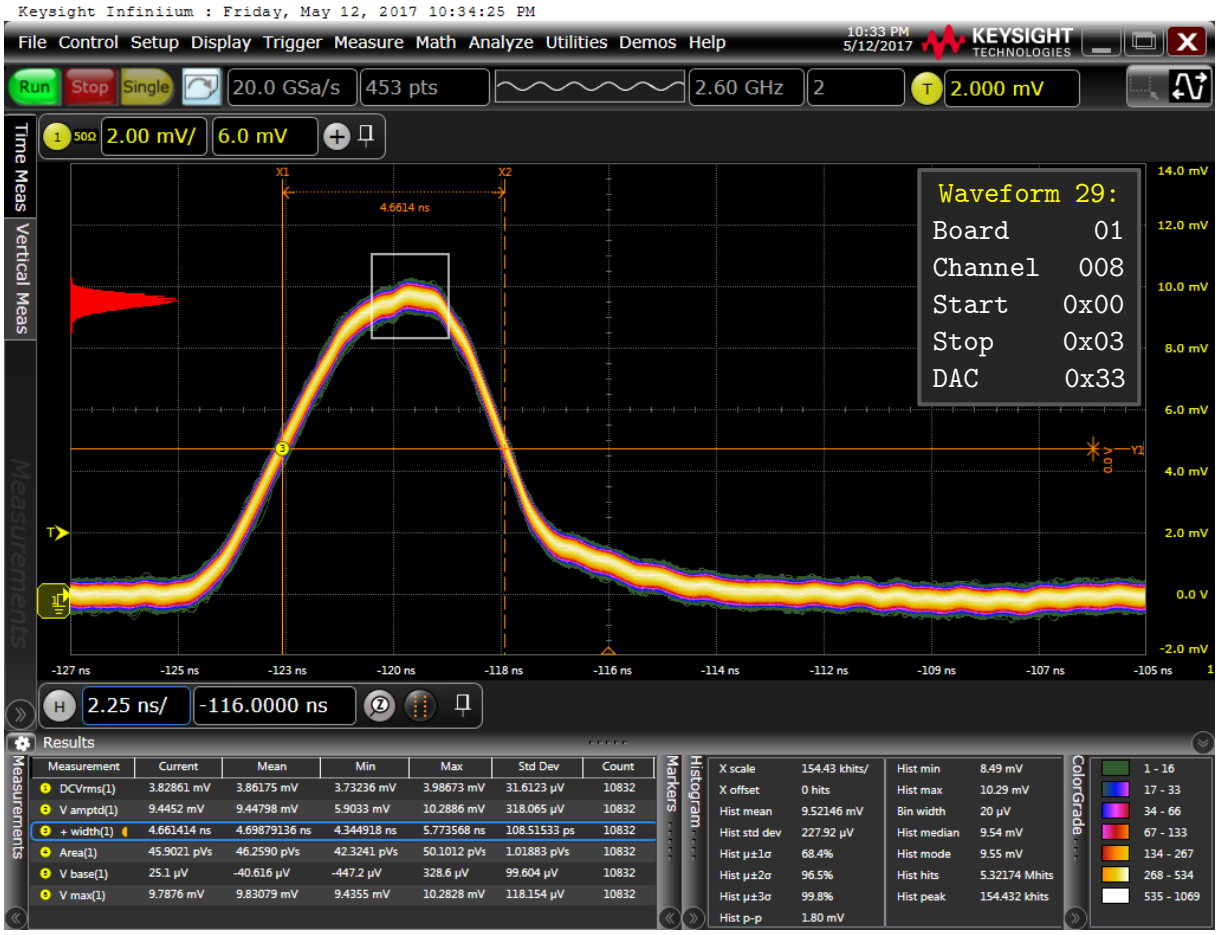
Keysight Infiniium : Friday, May 12, 2017 10:27:01 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:26 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6289 ns
Waveform 24:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x88
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.98356 mV 3.01710 mV 2.89734 mV 3.14159 mV 30.5437 µV 10240
V amptd(1) 7.2863 mV 7.33707 mV 5.0599 mV 8.0622 mV 285.717 µV 10240
+ width(1) 4.628942 ns 4.59857828 ns 4.254134 ns 5.471299 ns 118.11799 ps 10240
Area(1) 35.2266 pVs 35.4885 pVs 30.9757 pVs 39.3606 pVs 1.04831 pVs 10240
V base(1) -109.9 µV -27.457 µV -346.0 µV 328.6 µV 94.802 µV 10240
V max(1) 7.6909 mV 7.75215 mV 7.3019 mV 8.1689 mV 115.077 µV 10240
X scale 167.81 khits/
X offset 0 hits
Hist mean 7.25571 mV
Hist std dev 345.46 µV
Hist µ±1σ 66.7%
Hist µ±2σ 96.6%
Hist µ±3σ 99.9%
Hist p-p 2.03 mV
Hist min 6.15 mV
Hist max 8.18 mV
Bin width 20 µV
Hist median 7.30 mV
Hist mode 7.43 mV
Hist hits 8.28207 Mhits
Hist peak 167.811 khits
1 - 16
17 - 32
33 - 64
65 - 128
129 - 257
258 - 514
515 - 1029

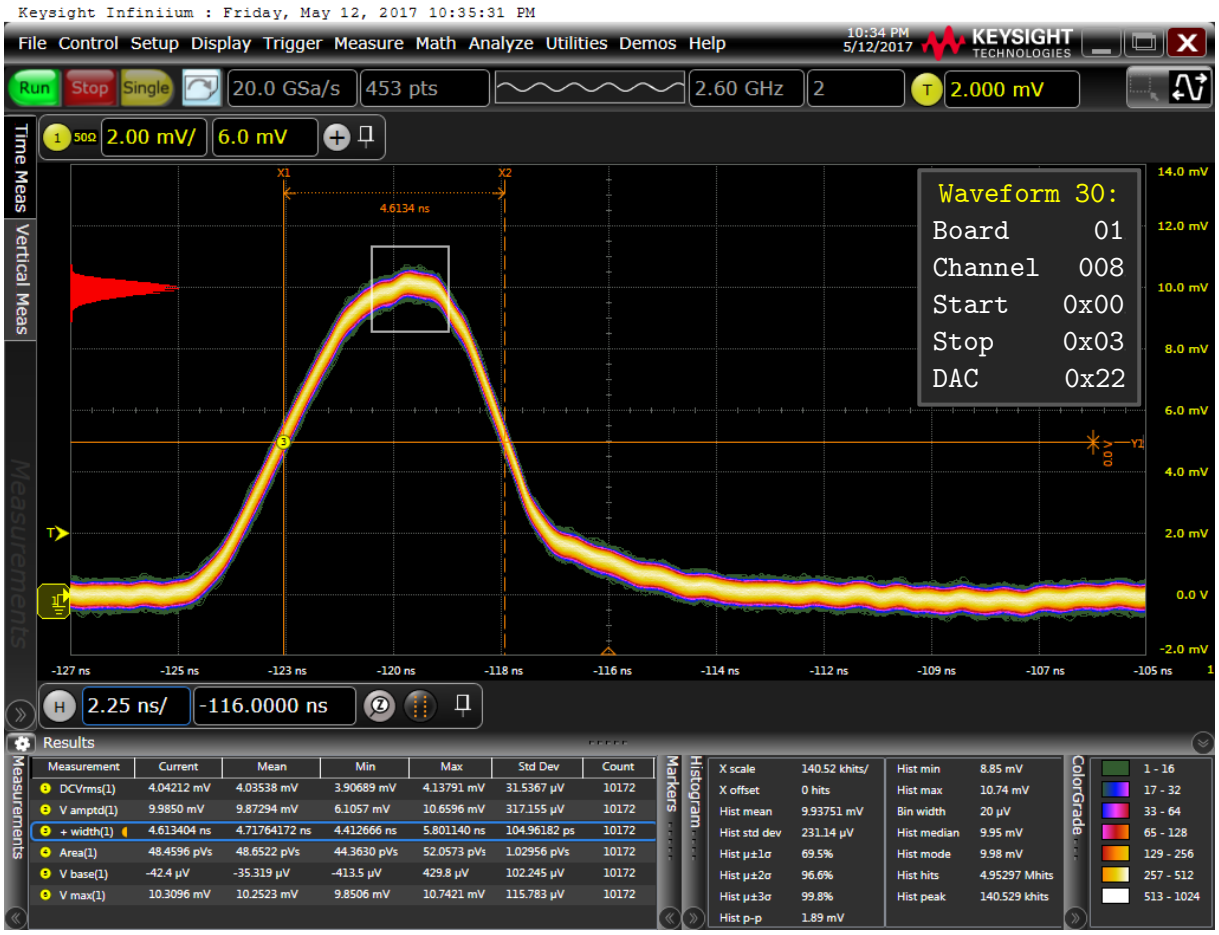

Keysight Infiniium : Friday, May 12, 2017 10:27:57 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:27 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
Waveform 25:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x77
4.5841 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.17478 mV 3.18614 mV 3.06965 mV 3.31262 mV 31.6256 µV 10192
V amptd(1) 7.6574 mV 7.75445 mV 4.8576 mV 8.5344 mV 299.766 µV 10192
+ width(1) 4.584078 ns 4.62196835 ns 4.280513 ns 5.520441 ns 117.34007 ps 10192
Area(1) 36.5714 pVs 37.6229 pVs 33.3393 pVs 41.6466 pVs 1.06719 pVs 10192
V base(1) 92.5 µV -29.610 µV -379.7 µV 328.6 µV 96.033 µV 10192
V max(1) 8.0669 mV 8.17330 mV 7.7865 mV 8.6254 mV 117.385 µV 10192
X scale 159.65 khits/
X offset 0 hits
Hist mean 7.65212 mV
Hist std dev 369.42 µV
Hist µ±1σ 66.6%
Hist µ±2σ 96.6%
Hist µ±3σ 99.9%
Hist p-p 2.17 mV
Hist min 6.46 mV
Hist max 8.63 mV
Bin width 20 µV
Hist median 7.71 mV
Hist mode 7.85 mV
Hist hits 8.30319 Mhits
Hist peak 159.657 khits
1 - 15
16 - 31
32 - 63
64 - 127
128 - 254
255 - 509
510 - 1018


Keysight Infiniium : Friday, May 12, 2017 10:29:00 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:28 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
Waveform 26:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x66
4.6433 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.35049 mV 3.35534 mV 3.22289 mV 3.48413 mV 30.7112 µV 10512
V amptd(1) 8.0284 mV 8.17969 mV 5.1949 mV 8.9730 mV 303.360 µV 10512
+ width(1) 4.643301 ns 4.64055953 ns 4.302958 ns 5.374144 ns 114.93169 ps 10512
Area(1) 39.3736 pVs 39.7980 pVs 36.0035 pVs 44.1052 pVs 1.02345 pVs 10512
V base(1) 58.8 µV -31.617 µV -379.7 µV 328.6 µV 95.143 µV 10512
V max(1) 8.4965 mV 8.59099 mV 8.1787 mV 9.0726 mV 116.407 µV 10512
X scale 162.77 khits/
X offset 0 hits
Hist mean 8.04878 mV
Hist std dev 391.60 µV
Hist µ±1σ 66.7%
Hist µ±2σ 96.3%
Hist µ±3σ 100%
Hist p-p 2.23 mV
Hist min 6.85 mV
Hist max 9.09 mV
Bin width 20 µV
Hist median 8.12 mV
Hist mode 8.30 mV
Hist hits 8.61221 Mhits
Hist peak 162.771 khits
1 - 16
17 - 33
34 - 66
67 - 132
133 - 265
266 - 530
531 - 1060

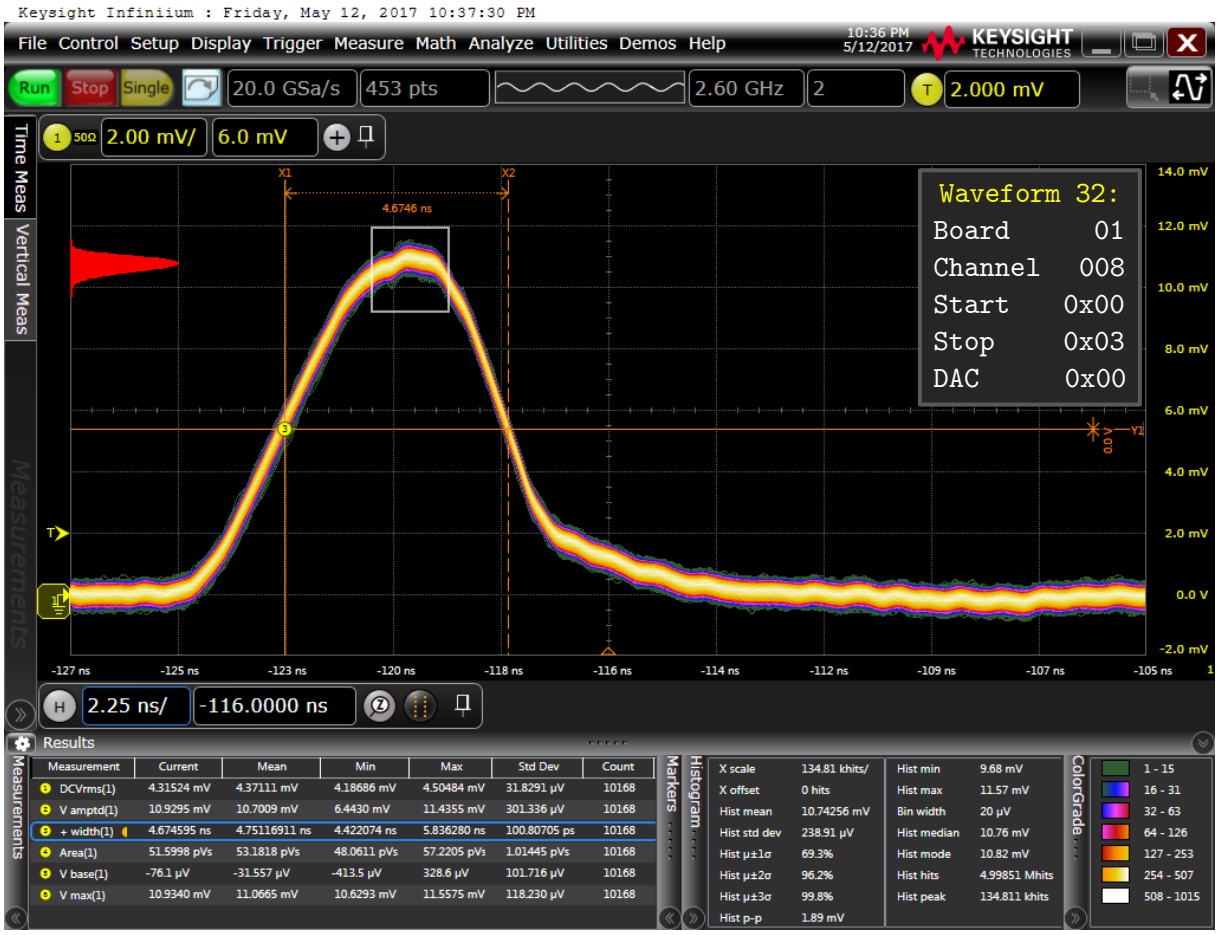
Keysight Infiniium : Friday, May 12, 2017 10:30:12 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:29 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2.000 mV
2.00 mV/
6.0 mV
4.7403 ns
Waveform 27:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x55
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.51628 mV 3.52118 mV 3.39258 mV 3.64480 mV 30.7421 µV 10136
V amptd(1) 8.0284 mV 8.60592 mV 5.8358 mV 9.3440 mV 302.155 µV 10136
+ width(1) 4.740290 ns 4.65785627 ns 4.331191 ns 5.510638 ns 109.31973 ps 10136
Area(1) 41.4479 pVs 41.8048 pVs 37.7016 pVs 45.2537 pVs 1.02680 pVs 10136
V base(1) -8.7 µV -40.376 µV -379.7 µV 294.9 µV 96.929 µV 10136
V max(1) 9.0341 mV 8.99578 mV 8.6201 mV 9.5259 mV 116.607 µV 10136
X scale 152.19 khits/
X offset 0 hits
Hist mean 8.51470 mV
Hist std dev 344.70 µV
Hist µ±1σ 66.6%
Hist µ±2σ 96.6%
Hist µ±3σ 99.9%
Hist p-p 2.31 mV
Hist min 7.23 mV
Hist max 9.54 mV
Bin width 20 µV
Hist median 8.57 mV
Hist mode 8.73 mV
Hist hits 7.35172 Mhits
Hist peak 152.197 khits
1 - 16
17 - 32
33 - 64
65 - 128
129 - 256
257 - 512
513 - 1024

Keysight Infiniium : Friday, May 12, 2017 10:31:14 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:30 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2.000 mV
2.00 mV/
6.0 mV
4.6260 ns
Waveform 28:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x44
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.67370 mV 3.69261 mV 3.56948 mV 3.83186 mV 30.8634 µV 10485
V amptd(1) 9.1079 mV 9.02112 mV 6.0045 mV 9.7826 mV 309.254 µV 10485
+ width(1) 4.625963 ns 4.68038210 ns 4.329397 ns 5.541393 ns 110.35046 ps 10485
Area(1) 43.6663 pVs 44.1263 pVs 40.1334 pVs 48.1352 pVs 1.00927 pVs 10485
V base(1) -8.7 µV -38.088 µV -480.9 µV 328.6 µV 98.342 µV 10485
V max(1) 9.2921 mV 9.41851 mV 9.0104 mV 9.8782 mV 117.318 µV 10485
X scale 152.82 khits/
X offset 0 hits
Hist mean 8.91991 mV
Hist std dev 358.03 µV
Hist µ±1σ 66.7%
Hist µ±2σ 96.8%
Hist µ±3σ 100%
Hist p-p 2.16 mV
Hist min 7.73 mV
Hist max 9.88 mV
Bin width 20 µV
Hist median 8.98 mV
Hist mode 9.15 mV
Hist hits 7.64220 Mhits
Hist peak 152.828 khits
1 - 16
17 - 32
33 - 65
66 - 130
131 - 260
261 - 521
522 - 1042


Keysight Infiniium : Friday, May 12, 2017 10:34:25 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:33 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6614 ns
Waveform 29:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x33
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.82861 mV 3.86175 mV 3.73236 mV 3.98673 mV 31.6123 µV 10832
V amptd(1) 9.4452 mV 9.44798 mV 5.9033 mV 10.2886 mV 318.065 µV 10832
+ width(1) 4.661414 ns 4.69879136 ns 4.344918 ns 5.773568 ns 108.51533 ps 10832
Area(1) 45.9021 pVs 46.2590 pVs 42.3241 pVs 50.1012 pVs 1.01883 pVs 10832
V base(1) 25.1 µV -40.616 µV -447.2 µV 328.6 µV 99.604 µV 10832
V max(1) 9.7876 mV 9.83079 mV 9.4355 mV 10.2828 mV 118.154 µV 10832
X scale 154.43 khits/
X offset 0 hits
Hist mean 9.52146 mV
Hist std dev 227.92 µV
Hist µ±1σ 68.4%
Hist µ±2σ 96.5%
Hist µ±3σ 99.8%
Hist p-p 1.80 mV
Hist min 8.49 mV
Hist max 10.29 mV
Bin width 20 µV
Hist median 9.54 mV
Hist mode 9.55 mV
Hist hits 5.32174 Mhits
Hist peak 154.432 khits
1 - 16
17 - 33
34 - 66
67 - 133
134 - 267
268 - 534
535 - 1069


Keysight Infiniium : Friday, May 12, 2017 10:35:31 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:34 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6134 ns
Waveform 30:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x22
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.04212 mV 4.03538 mV 3.90689 mV 4.13791 mV 31.5367 µV 10172
V amptd(1) 9.9850 mV 9.87294 mV 6.1057 mV 10.6596 mV 317.155 µV 10172
+ width(1) 4.613404 ns 4.71764172 ns 4.412666 ns 5.801140 ns 104.96182 ps 10172
Area(1) 48.4596 pVs 48.6522 pVs 44.3630 pVs 52.0573 pVs 1.02956 pVs 10172
V base(1) -42.4 µV -35.319 µV -413.5 µV 429.8 µV 102.245 µV 10172
V max(1) 10.3096 mV 10.2523 mV 9.8506 mV 10.7421 mV 115.783 µV 10172
X scale 140.52 khits/
X offset 0 hits
Hist mean 9.93751 mV
Hist std dev 231.14 µV
Hist µ±1σ 69.5%
Hist µ±2σ 96.6%
Hist µ±3σ 99.8%
Hist p-p 1.89 mV
Hist min 8.85 mV
Hist max 10.74 mV
Bin width 20 µV
Hist median 9.95 mV
Hist mode 9.98 mV
Hist hits 4.95297 Mhits
Hist peak 140.529 khits
1 - 16
17 - 32
33 - 64
65 - 128
129 - 256
257 - 512
513 - 1024

Keysight Infiniium : Friday, May 12, 2017 10:36:34 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:35 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6182 ns
Waveform 31:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x11
2.25 ns/
-116.0000 ns
Results
Measurements
Markers
Histogram
ColorGrade


Keysight Infiniium : Friday, May 12, 2017 10:37:30 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:36 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
4.6746 ns
Waveform 32:
Board 01
Channel 008
Start 0x00
Stop 0x03
DAC 0x00
2.25 ns/
-116.0000 ns
Results
Measurements
Markers
Histogram
ColorGrade

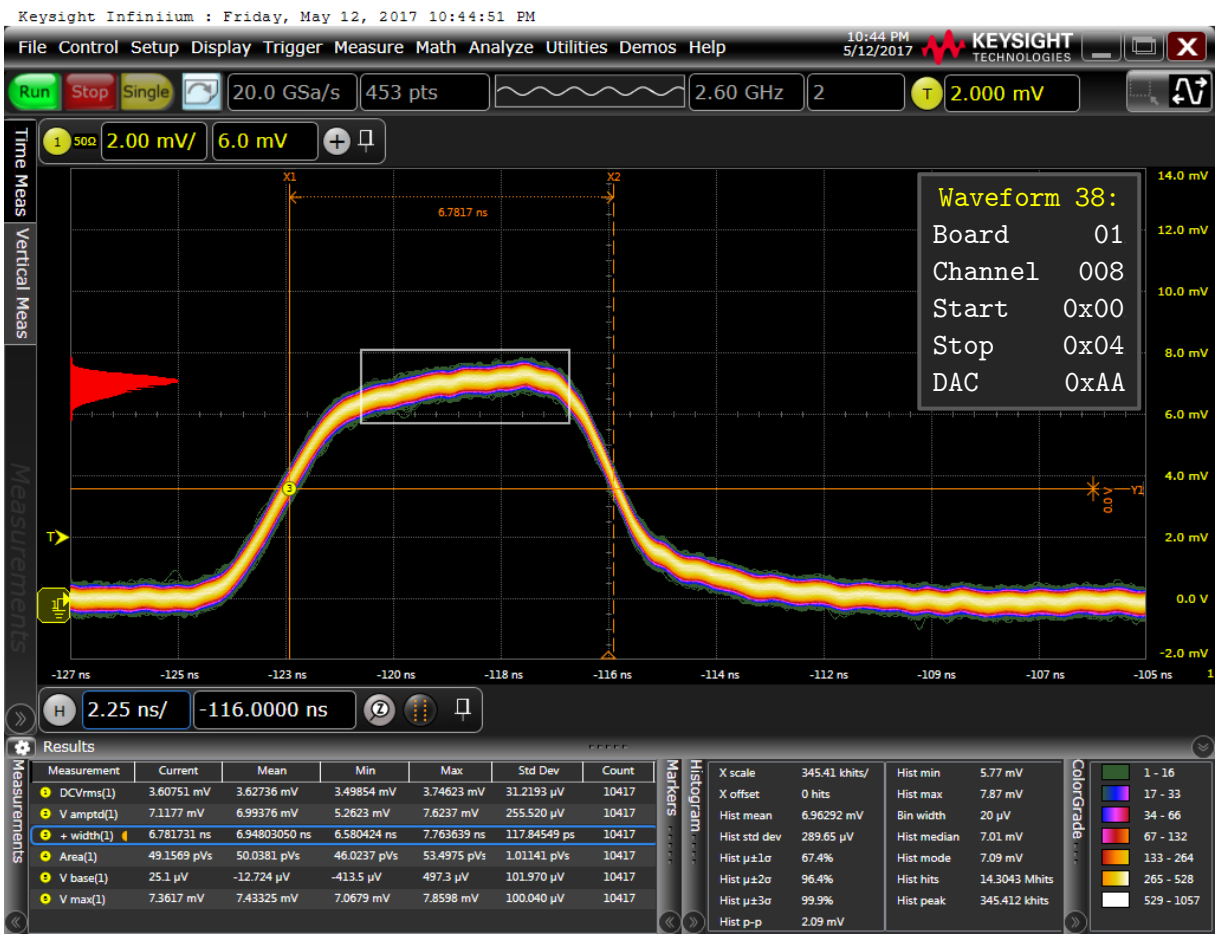
Keysight Infiniium : Friday, May 12, 2017 10:38:53 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:38 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.5420 ns
Waveform 33:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0xFF
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.42944 mV 2.43228 mV 2.31371 mV 2.55195 mV 29.5342 µV 10126
V amptd(1) 5.0599 mV 4.74494 mV 3.2046 mV 5.3298 mV 214.676 µV 10126
+ width(1) 6.542002 ns 6.58045941 ns 6.154966 ns 7.239430 ns 112.47823 ps 10126
Area(1) 32.8092 pVs 32.3877 pVs 28.3707 pVs 36.1432 pVs 987.409 fVs 10126
V base(1) -109.9 µV -1.887 µV -346.0 µV 362.4 µV 95.098 µV 10126
V max(1) 5.1222 mV 5.17515 mV 4.8367 mV 5.6076 mV 96.964 µV 10126
X scale 373.79 khits/
X offset 0 hits
Hist mean 4.65216 mV
Hist std dev 325.04 µV
Hist µ±1σ 67.9%
Hist µ±2σ 96.1%
Hist µ±3σ 100%
Hist p-p 1.77 mV
Hist min 3.85 mV
Hist max 5.62 mV
Bin width 20 µV
Hist median 4.70 mV
Hist mode 4.85 mV
Hist hits 16.9384 Mhits
Hist peak 373.799 khits
1 - 15
16 - 31
32 - 62
63 - 125
126 - 250
251 - 501
502 - 1003

Keysight Infiniium : Friday, May 12, 2017 10:40:12 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:39 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.7056 ns
Waveform 34:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0xEE
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.64627 mV 2.65278 mV 2.54473 mV 2.77166 mV 30.0299 µV 10312
V amptd(1) 5.2286 mV 5.18698 mV 3.4745 mV 5.7683 mV 223.358 µV 10312
+ width(1) 6.705566 ns 6.60222895 ns 6.229017 ns 7.477746 ns 110.61426 ps 10312
Area(1) 35.5742 pVs 35.5041 pVs 31.8849 pVs 39.7332 pVs 1.00012 pVs 10312
V base(1) -8.7 µV -2.932 µV -379.7 µV 429.8 µV 94.242 µV 10312
V max(1) 5.5683 mV 5.61794 mV 5.3177 mV 6.0261 mV 97.149 µV 10312
X scale 363.92 khits/
X offset 0 hits
Hist mean 5.07448 mV
Hist std dev 352.39 µV
Hist µ±1σ 68.2%
Hist µ±2σ 95.6%
Hist µ±3σ 100%
Hist p-p 1.97 mV
Hist min 4.07 mV
Hist max 6.04 mV
Bin width 20 µV
Hist median 5.13 mV
Hist mode 5.30 mV
Hist hits 17.0085 Mhits
Hist peak 363.926 khits
1 - 16
17 - 33
34 - 66
67 - 133
134 - 267
268 - 534
535 - 1069

Keysight Infiniium : Friday, May 12, 2017 10:41:24 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:40 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.7366 ns
Waveform 35:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0xDD
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.88321 mV 2.88953 mV 2.76679 mV 2.99950 mV 30.9477 µV 10123
V amptd(1) 5.2961 mV 5.63959 mV 3.7106 mV 6.2406 mV 224.952 µV 10123
+ width(1) 6.736629 ns 6.68348769 ns 6.227061 ns 7.371059 ns 106.70696 ps 10123
Area(1) 38.8951 pVs 38.9512 pVs 34.5986 pVs 42.2019 pVs 1.01444 pVs 10123
V base(1) 126.2 µV -5.354 µV -379.7 µV 396.1 µV 95.447 µV 10123
V max(1) 5.9594 mV 6.06592 mV 5.6906 mV 6.4485 mV 98.028 µV 10123
X scale 355.22 khits/
X offset 0 hits
Hist mean 5.56012 mV
Hist std dev 321.21 µV
Hist µ±1σ 67.8%
Hist µ±2σ 96.1%
Hist µ±3σ 99.8%
Hist p-p 2.16 mV
Hist min 4.30 mV
Hist max 6.46 mV
Bin width 20 µV
Hist median 5.62 mV
Hist mode 5.73 mV
Hist hits 15.5742 Mhits
Hist peak 355.228 khits
1 - 16
17 - 32
33 - 64
65 - 128
129 - 256
257 - 512
513 - 1024

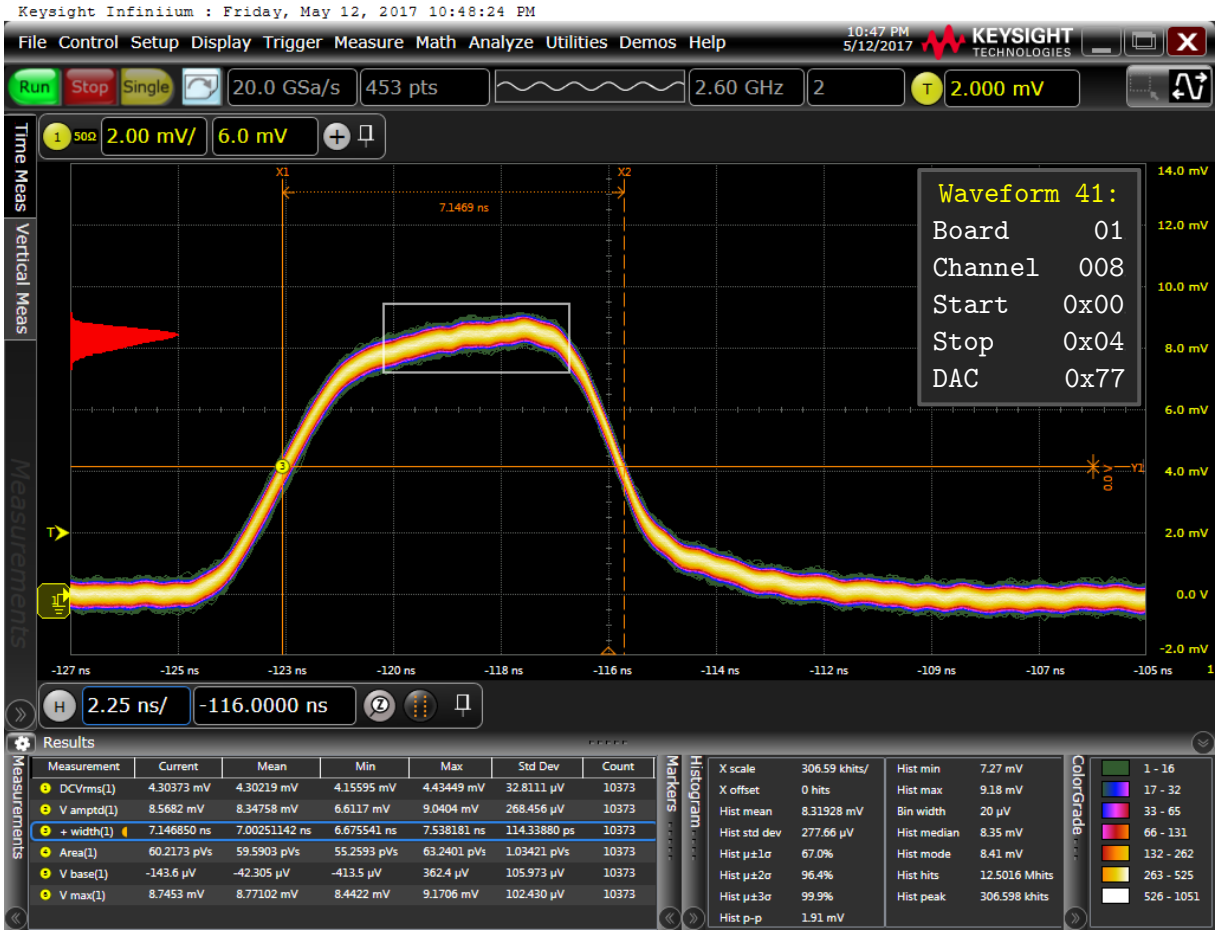
Keysight Infiniium : Friday, May 12, 2017 10:42:36 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:41 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.8478 ns
Waveform 36:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0xCC
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.13752 mV 3.15097 mV 3.02150 mV 3.27519 mV 31.1842 µV 10123
V amptd(1) 5.9707 mV 6.09172 mV 4.6889 mV 6.8815 mV 236.538 µV 10123
+ width(1) 6.847787 ns 6.85135759 ns 6.466530 ns 7.392751 ns 115.82833 ps 10123
Area(1) 41.6918 pVs 42.9065 pVs 38.5625 pVs 46.9276 pVs 1.01874 pVs 10123
V base(1) 25.1 µV -13.728 µV -379.7 µV 396.1 µV 97.790 µV 10123
V max(1) 6.4438 mV 6.52190 mV 6.1766 mV 6.9221 mV 99.127 µV 10123
X scale 349.84 khits/
X offset 0 hits
Hist mean 6.00930 mV
Hist std dev 326.22 µV
Hist µ±1σ 66.9%
Hist µ±2σ 96.0%
Hist µ±3σ 100%
Hist p-p 1.89 mV
Hist min 5.04 mV
Hist max 6.93 mV
Bin width 20 µV
Hist median 6.07 mV
Hist mode 6.21 mV
Hist hits 15.4092 Mhits
Hist peak 349.847 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 253
254 - 506
507 - 1012

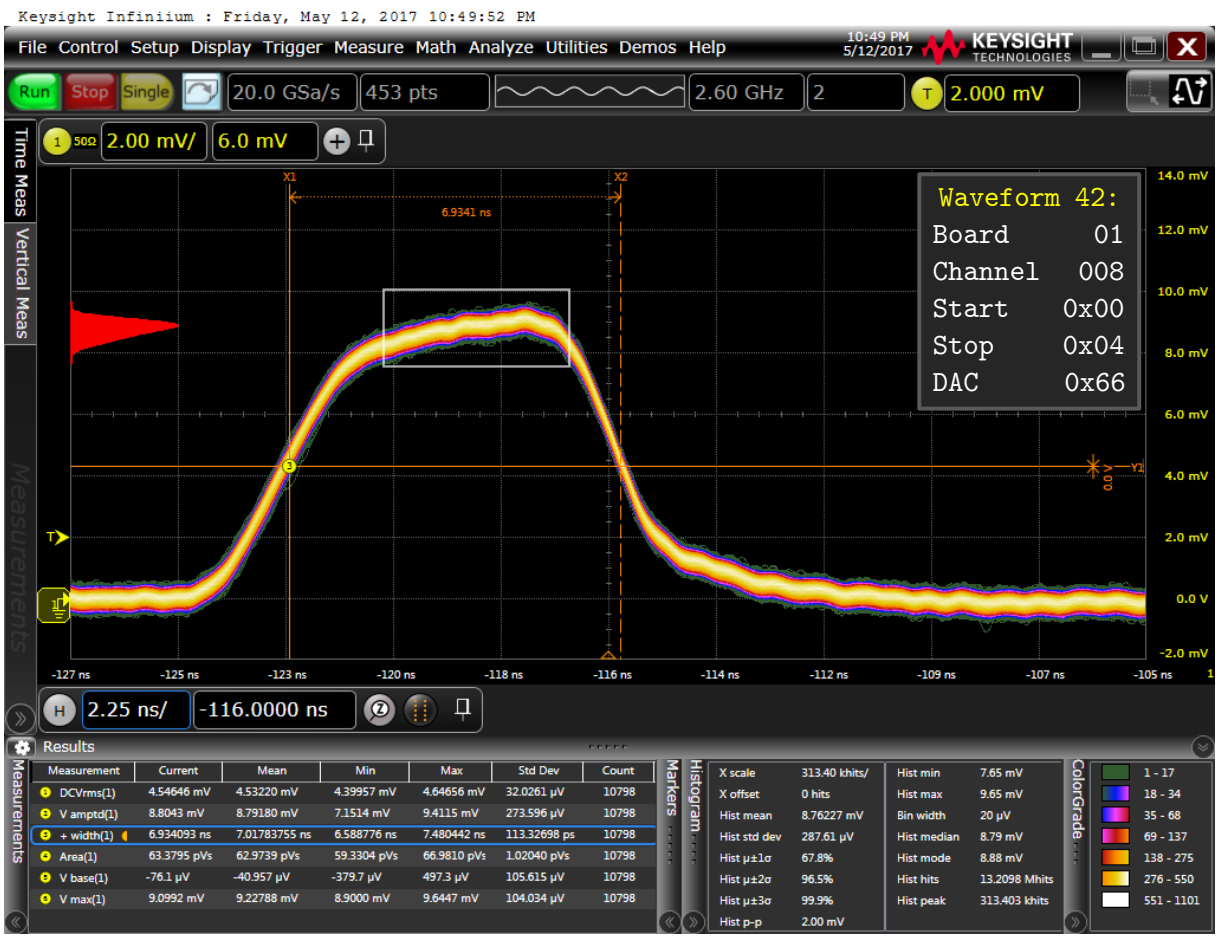
Keysight Infiniium : Friday, May 12, 2017 10:43:52 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:43 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.8956 ns
Waveform 37:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0xBB
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.39549 mV 3.39028 mV 3.25562 mV 3.51595 mV 30.8993 µV 10178
V amptd(1) 6.5779 mV 6.54134 mV 5.0262 mV 7.1514 mV 244.664 µV 10178
+ width(1) 6.895585 ns 6.90004455 ns 6.495982 ns 7.502376 ns 115.46671 ps 10178
Area(1) 47.6882 pVs 46.5382 pVs 42.3537 pVs 50.4481 pVs 1.01060 pVs 10178
V base(1) 92.5 µV -8.477 µV -447.2 µV 396.1 µV 101.009 µV 10178
V max(1) 6.8868 mV 6.97691 mV 6.6346 mV 7.4381 mV 100.301 µV 10178
X scale 347.18 khits/
X offset 0 hits
Hist mean 6.52203 mV
Hist std dev 277.39 µV
Hist µ±1σ 67.0%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.88 mV
Hist min 5.57 mV
Hist max 7.45 mV
Bin width 20 µV
Hist median 6.55 mV
Hist mode 6.63 mV
Hist hits 14.0119 Mhits
Hist peak 347.184 khits
1 - 16
17 - 32
33 - 65
66 - 131
132 - 263
264 - 526
527 - 1052

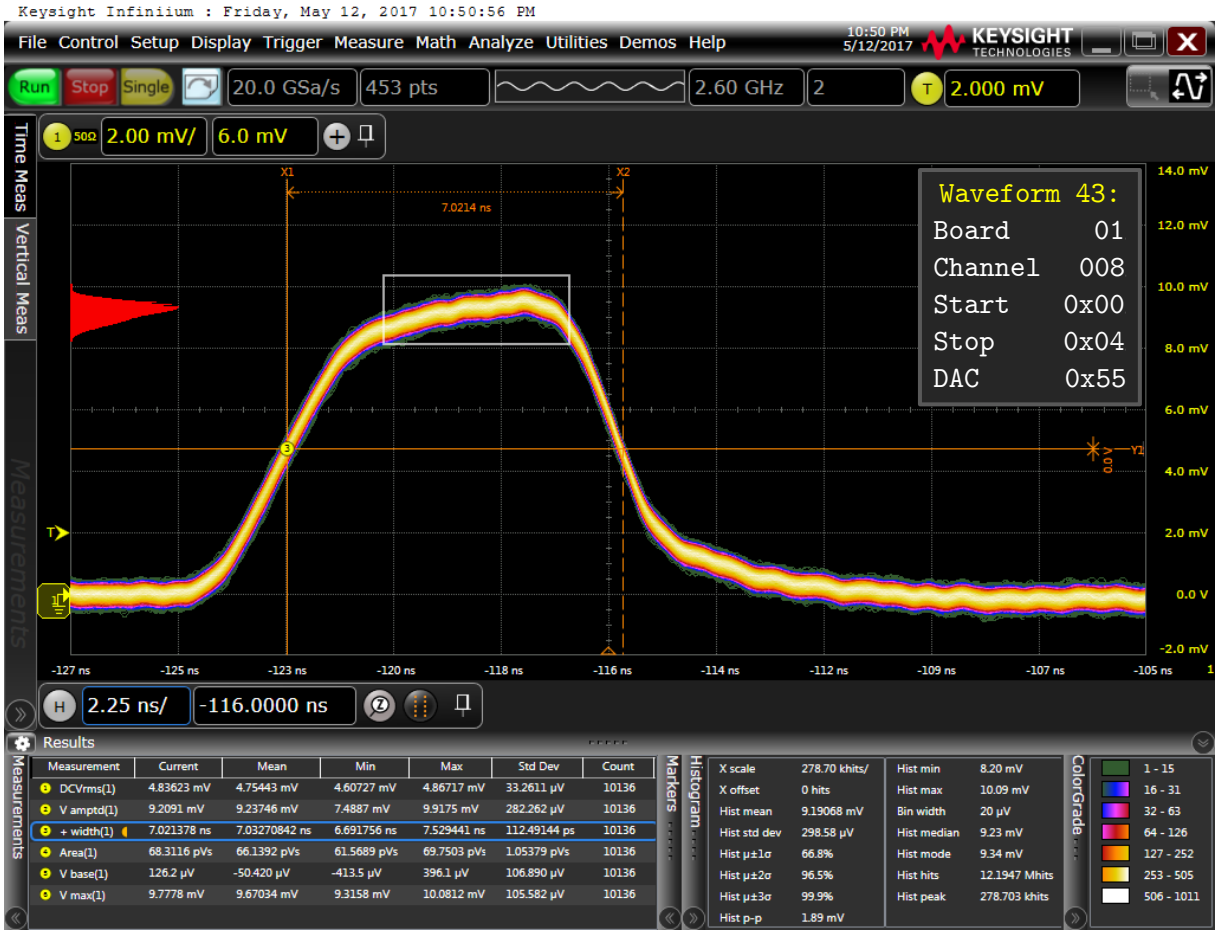

Keysight Infiniium : Friday, May 12, 2017 10:44:51 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:44 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.7817 ns
Waveform 38:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0xAA
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.60751 mV 3.62736 mV 3.49854 mV 3.74623 mV 31.2193 µV 10417
V amptd(1) 7.1177 mV 6.99376 mV 5.2623 mV 7.6237 mV 255.520 µV 10417
+ width(1) 6.781731 ns 6.94803050 ns 6.580424 ns 7.763639 ns 117.84549 ps 10417
Area(1) 49.1569 pVs 50.0381 pVs 46.0237 pVs 53.4975 pVs 1.01141 pVs 10417
V base(1) 25.1 µV -12.724 µV -413.5 µV 497.3 µV 101.970 µV 10417
V max(1) 7.3617 mV 7.43325 mV 7.0679 mV 7.8598 mV 100.040 µV 10417
X scale 345.41 khits/
X offset 0 hits
Hist mean 6.96292 mV
Hist std dev 289.65 µV
Hist µ±1σ 67.4%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 2.09 mV
Hist min 5.77 mV
Hist max 7.87 mV
Bin width 20 µV
Hist median 7.01 mV
Hist mode 7.09 mV
Hist hits 14.3043 Mhits
Hist peak 345.412 khits
1 - 16
17 - 33
34 - 66
67 - 132
133 - 264
265 - 528
529 - 1057

Keysight Infiniium : Friday, May 12, 2017 10:46:16 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:45 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.0973 ns
Waveform 39:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x99
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.83824 mV 3.85270 mV 3.73371 mV 3.96415 mV 31.6441 µV 10464
V amptd(1) 7.3201 mV 7.44281 mV 6.1057 mV 8.0622 mV 253.935 µV 10464
+ width(1) 7.097338 ns 6.96677035 ns 6.625052 ns 7.495192 ns 115.00939 ps 10464
Area(1) 53.0403 pVs 53.2695 pVs 49.1010 pVs 57.0983 pVs 1.01746 pVs 10464
V base(1) -42.4 µV -21.271 µV -379.7 µV 396.1 µV 101.732 µV 10464
V max(1) 7.9563 mV 7.87561 mV 7.5169 mV 8.3481 mV 101.544 µV 10464
X scale 337.63 khits/
X offset 0 hits
Hist mean 7.39299 mV
Hist std dev 301.62 µV
Hist µ±1σ 66.4%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 2.16 mV
Hist min 6.20 mV
Hist max 8.35 mV
Bin width 20 µV
Hist median 7.43 mV
Hist mode 7.55 mV
Hist hits 14.3609 Mhits
Hist peak 337.634 khits
1 - 16
17 - 33
34 - 67
68 - 134
135 - 269
270 - 539
540 - 1079

Keysight Infiniium : Friday, May 12, 2017 10:47:13 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:46 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.9565 ns
Waveform 40:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x88
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.08850 mV 4.08083 mV 3.95832 mV 4.20073 mV 32.5497 µV 10100
V amptd(1) 7.7249 mV 7.89106 mV 6.2406 mV 8.5682 mV 265.805 µV 10100
+ width(1) 6.956523 ns 6.98683024 ns 6.628311 ns 7.527004 ns 114.69210 ps 10100
Area(1) 57.4451 pVs 56.5208 pVs 52.8087 pVs 60.3247 pVs 1.03324 pVs 10100
V base(1) 193.7 µV -24.490 µV -413.5 µV 429.8 µV 103.951 µV 10100
V max(1) 8.2596 mV 8.33250 mV 7.9805 mV 8.7579 mV 102.113 µV 10100
X scale 303.23 khits/
X offset 0 hits
Hist mean 7.82624 mV
Hist std dev 318.12 µV
Hist µ±1σ 66.3%
Hist µ±2σ 96.7%
Hist µ±3σ 99.9%
Hist p-p 2.03 mV
Hist min 6.73 mV
Hist max 8.76 mV
Bin width 20 µV
Hist median 7.87 mV
Hist mode 7.96 mV
Hist hits 13.6373 Mhits
Hist peak 303.239 khits
1 - 16
17 - 32
33 - 65
66 - 130
131 - 260
261 - 521
522 - 1042

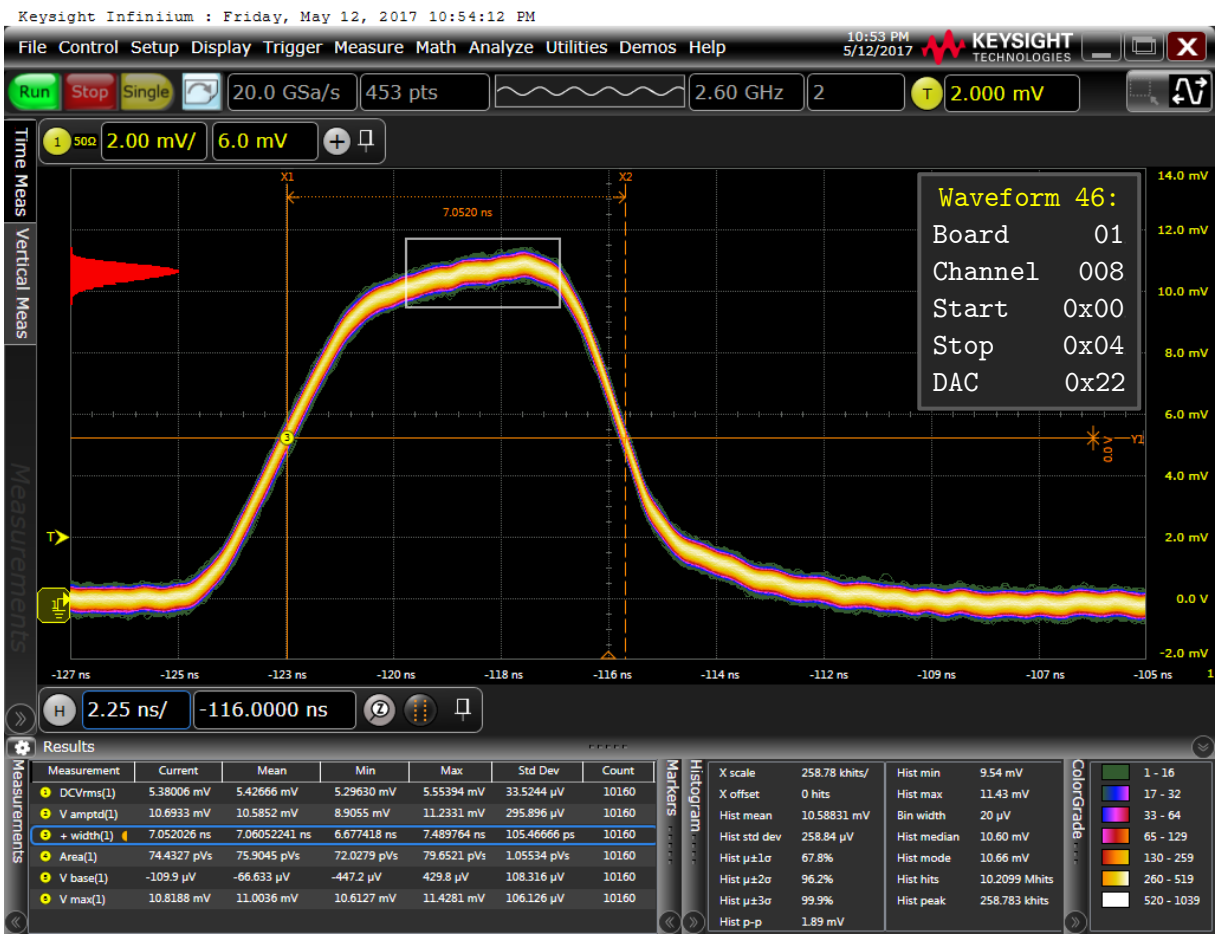

Keysight Infiniium : Friday, May 12, 2017 10:48:24 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:47 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.1469 ns
Waveform 41:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x77
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.30373 mV 4.30219 mV 4.15595 mV 4.43449 mV 32.8111 µV 10373
V amptd(1) 8.5682 mV 8.34758 mV 6.6117 mV 9.0404 mV 268.456 µV 10373
+ width(1) 7.146850 ns 7.00251142 ns 6.675541 ns 7.538181 ns 114.33880 ps 10373
Area(1) 60.2173 pVs 59.5903 pVs 55.2593 pVs 63.2401 pVs 1.03421 pVs 10373
V base(1) -143.6 µV -42.305 µV -413.5 µV 362.4 µV 105.973 µV 10373
V max(1) 8.7453 mV 8.77102 mV 8.4422 mV 9.1706 mV 102.430 µV 10373
X scale 306.59 khits/
X offset 0 hits
Hist mean 8.31928 mV
Hist std dev 277.66 µV
Hist µ±1σ 67.0%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.91 mV
Hist min 7.27 mV
Hist max 9.18 mV
Bin width 20 µV
Hist median 8.35 mV
Hist mode 8.41 mV
Hist hits 12.5016 Mhits
Hist peak 306.598 khits
1 - 16
17 - 32
33 - 65
66 - 131
132 - 262
263 - 525
526 - 1051


Keysight Infiniium : Friday, May 12, 2017 10:49:52 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:49 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.9341 ns
Waveform 42:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x66
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.54646 mV 4.53220 mV 4.39957 mV 4.64656 mV 32.0261 µV 10798
V amptd(1) 8.8043 mV 8.79180 mV 7.1514 mV 9.4115 mV 273.596 µV 10798
+ width(1) 6.934093 ns 7.01783755 ns 6.588776 ns 7.480442 ns 113.32698 ps 10798
Area(1) 63.3795 pVs 62.9739 pVs 59.3304 pVs 66.9810 pVs 1.02040 pVs 10798
V base(1) -76.1 µV -40.957 µV -379.7 µV 497.3 µV 105.615 µV 10798
V max(1) 9.0992 mV 9.22788 mV 8.9000 mV 9.6447 mV 104.034 µV 10798
X scale 313.40 khits/
X offset 0 hits
Hist mean 8.76227 mV
Hist std dev 287.61 µV
Hist µ±1σ 67.8%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 2.00 mV
Hist min 7.65 mV
Hist max 9.65 mV
Bin width 20 µV
Hist median 8.79 mV
Hist mode 8.88 mV
Hist hits 13.2098 Mhits
Hist peak 313.403 khits
1 - 17
18 - 34
35 - 68
69 - 137
138 - 275
276 - 550
551 - 1101


Keysight Infiniium : Friday, May 12, 2017 10:50:56 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:50 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.0214 ns
Waveform 43:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x55
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.83623 mV 4.75443 mV 4.60727 mV 4.86717 mV 33.2611 µV 10136
V amptd(1) 9.2091 mV 9.23746 mV 7.4887 mV 9.9175 mV 282.262 µV 10136
+ width(1) 7.021378 ns 7.03270842 ns 6.691756 ns 7.529441 ns 112.49144 ps 10136
Area(1) 68.3116 pVs 66.1392 pVs 61.5689 pVs 69.7503 pVs 1.05379 pVs 10136
V base(1) 126.2 µV -50.420 µV -413.5 µV 396.1 µV 106.890 µV 10136
V max(1) 9.7778 mV 9.67034 mV 9.3158 mV 10.0812 mV 105.582 µV 10136
X scale 278.70 khits/
X offset 0 hits
Hist mean 9.19068 mV
Hist std dev 298.58 µV
Hist µ±1σ 66.8%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 1.89 mV
Hist min 8.20 mV
Hist max 10.09 mV
Bin width 20 µV
Hist median 9.23 mV
Hist mode 9.34 mV
Hist hits 12.1947 Mhits
Hist peak 278.703 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 252
253 - 505
506 - 1011

Keysight Infiniium : Friday, May 12, 2017 10:52:02 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:51 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.0026 ns
Waveform 44:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x44
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.93345 mV 4.98200 mV 4.82333 mV 5.10314 mV 33.4578 µV 10698
V amptd(1) 10.0524 mV 9.69386 mV 8.0622 mV 10.3898 mV 279.729 µV 10698
+ width(1) 7.002556 ns 7.03997123 ns 6.721544 ns 7.513183 ns 107.73053 ps 10698
Area(1) 68.0755 pVs 69.5088 pVs 64.9209 pVs 73.5849 pVs 1.04807 pVs 10698
V base(1) -177.3 µV -51.790 µV -413.5 µV 429.8 µV 105.751 µV 10698
V max(1) 9.9420 mV 10.1196 mV 9.7620 mV 10.5355 mV 105.847 µV 10698
X scale 291.66 khits/
X offset 0 hits
Hist mean 9.69293 mV
Hist std dev 265.00 µV
Hist µ±1σ 68.2%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.81 mV
Hist min 8.73 mV
Hist max 10.54 mV
Bin width 20 µV
Hist median 9.71 mV
Hist mode 9.76 mV
Hist hits 11.4451 Mhits
Hist peak 291.668 khits
1 - 17
18 - 34
35 - 69
70 - 138
139 - 277
278 - 555
556 - 1111

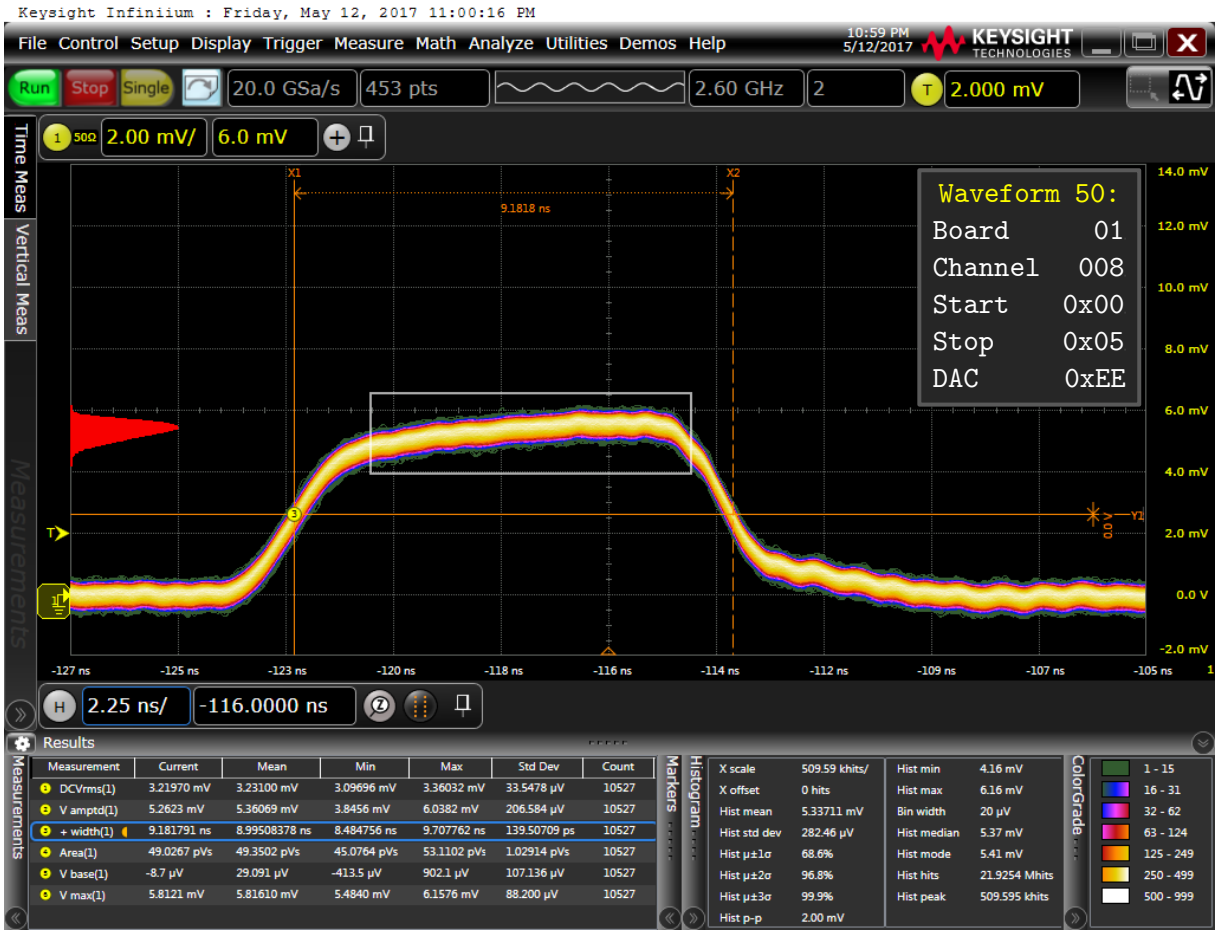
Keysight Infiniium : Friday, May 12, 2017 10:53:03 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:52 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.1114 ns
Waveform 45:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x33
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.22275 mV 5.20507 mV 5.08584 mV 5.31880 mV 32.7677 µV 10213
V amptd(1) 10.2886 mV 10.1512 mV 8.5344 mV 10.8283 mV 281.774 µV 10213
+ width(1) 7.111375 ns 7.05490457 ns 6.721882 ns 7.494606 ns 105.22942 ps 10213
Area(1) 73.0384 pVs 72.7079 pVs 68.8903 pVs 76.6918 pVs 1.03152 pVs 10213
V base(1) -42.4 µV -64.668 µV -480.9 µV 463.6 µV 104.284 µV 10213
V max(1) 10.6072 mV 10.5568 mV 10.2393 mV 11.0041 mV 103.936 µV 10213
X scale 272.03 khits/
X offset 0 hits
Hist mean 10.12554 mV
Hist std dev 271.39 µV
Hist µ±1σ 66.8%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 1.94 mV
Hist min 9.07 mV
Hist max 11.01 mV
Bin width 20 µV
Hist median 10.15 mV
Hist mode 10.24 mV
Hist hits 10.8015 Mhits
Hist peak 272.036 khits
1 - 17
18 - 34
35 - 68
69 - 137
138 - 275
276 - 550
551 - 1100


Keysight Infiniium : Friday, May 12, 2017 10:54:12 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:53 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.0520 ns
Waveform 46:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x22
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.38006 mV 5.42666 mV 5.29630 mV 5.55394 mV 33.5244 µV 10160
V amptd(1) 10.6933 mV 10.5852 mV 8.9055 mV 11.2331 mV 295.896 µV 10160
+ width(1) 7.052026 ns 7.06052241 ns 6.677418 ns 7.489764 ns 105.46666 ps 10160
Area(1) 74.4327 pVs 75.9045 pVs 72.0279 pVs 79.6521 pVs 1.05534 pVs 10160
V base(1) -109.9 µV -66.633 µV -447.2 µV 429.8 µV 108.316 µV 10160
V max(1) 10.8188 mV 11.0036 mV 10.6127 mV 11.4281 mV 106.126 µV 10160
X scale 258.78 khits/
X offset 0 hits
Hist mean 10.58831 mV
Hist std dev 258.84 µV
Hist µ±1σ 67.8%
Hist µ±2σ 96.2%
Hist µ±3σ 99.9%
Hist p-p 1.89 mV
Hist min 9.54 mV
Hist max 11.43 mV
Bin width 20 µV
Hist median 10.60 mV
Hist mode 10.66 mV
Hist hits 10.2099 Mhits
Hist peak 258.783 khits
1 - 16
17 - 32
33 - 64
65 - 129
130 - 259
260 - 519
520 - 1039

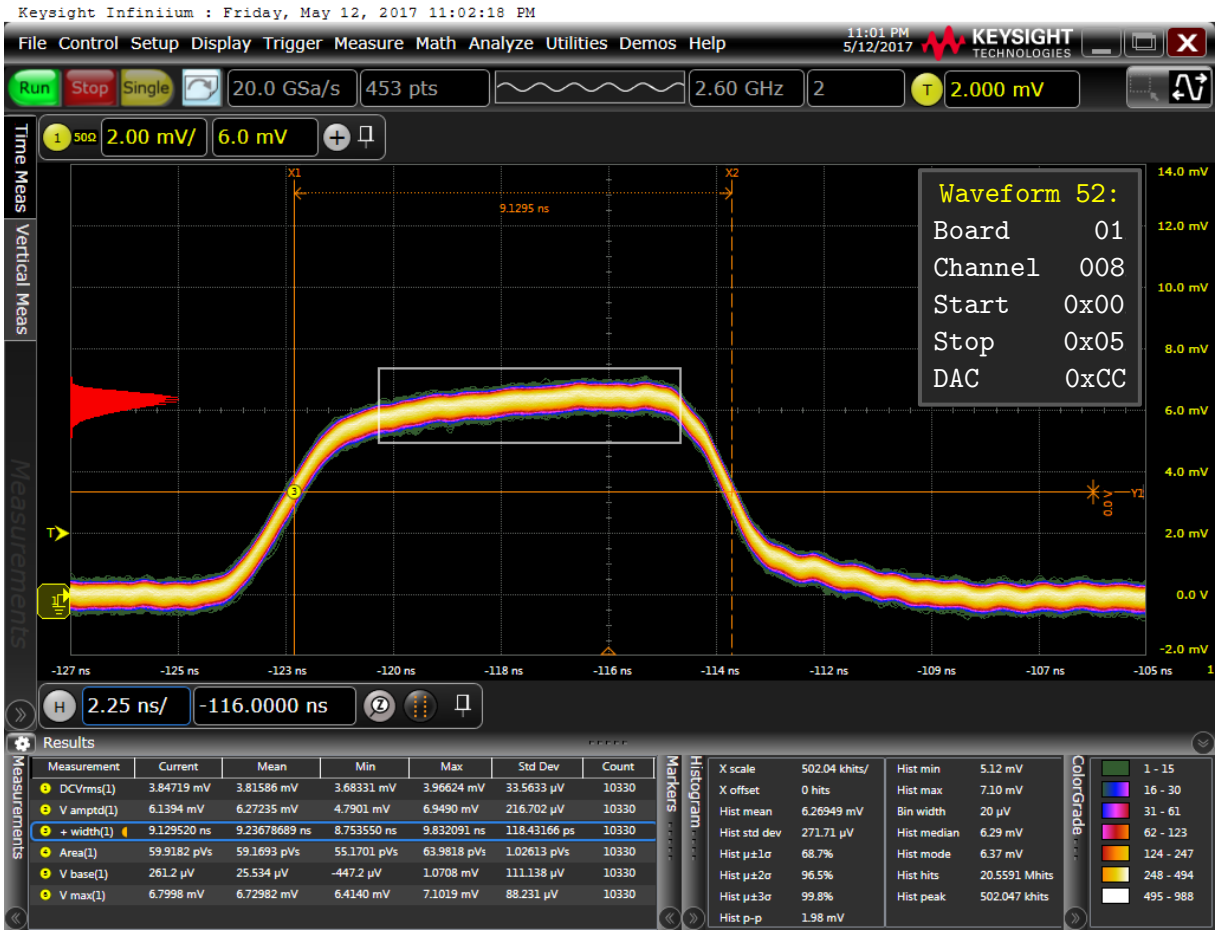
Keysight Infiniium : Friday, May 12, 2017 10:55:27 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:54 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
6.9738 ns
Waveform 47:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x11
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.63227 mV 5.65371 mV 5.50546 mV 5.78844 mV 33.7648 µV 10397
V amptd(1) 11.0981 mV 11.0214 mV 9.3103 mV 11.8740 mV 303.172 µV 10397
+ width(1) 6.973764 ns 7.07591519 ns 6.709263 ns 7.475610 ns 106.79405 ps 10397
Area(1) 79.2859 pVs 79.2795 pVs 75.3566 pVs 83.5739 pVs 1.05117 pVs 10397
V base(1) -76.1 µV -64.793 µV -413.5 µV 497.3 µV 111.844 µV 10397
V max(1) 11.3965 mV 11.4543 mV 11.1029 mV 12.0493 mV 108.549 µV 10397
X scale 256.65 khits/
X offset 0 hits
Hist mean 11.02596 mV
Hist std dev 268.62 µV
Hist µ±1σ 68.8%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 2.09 mV
Hist min 9.96 mV
Hist max 12.05 mV
Bin width 20 µV
Hist median 11.05 mV
Hist mode 11.12 mV
Hist hits 10.3751 Mhits
Hist peak 256.658 khits
1 - 16
17 - 33
34 - 66
67 - 133
134 - 267
268 - 535
536 - 1071

Keysight Infiniium : Friday, May 12, 2017 10:57:03 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:56 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
7.0207 ns
Waveform 48:
Board 01
Channel 008
Start 0x00
Stop 0x04
DAC 0x00
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.86744 mV 5.87551 mV 5.74970 mV 5.98969 mV 33.9108 µV 11173
V amptd(1) 11.6716 mV 11.4155 mV 9.2766 mV 12.2113 mV 324.108 µV 11173
+ width(1) 7.020667 ns 7.09249296 ns 6.727460 ns 7.580489 ns 113.53581 ps 11173
Area(1) 82.1413 pVs 82.6313 pVs 78.7071 pVs 86.6184 pVs 1.05296 pVs 11173
V base(1) -109.9 µV -46.313 µV -447.2 µV 767.2 µV 117.993 µV 11173
V max(1) 12.0306 mV 11.8924 mV 11.4964 mV 12.3595 mV 111.174 µV 11173
X scale 253.88 khits/
X offset 0 hits
Hist mean 11.50166 mV
Hist std dev 245.47 µV
Hist µ±1σ 69.0%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.95 mV
Hist min 10.41 mV
Hist max 12.37 mV
Bin width 20 µV
Hist median 11.51 mV
Hist mode 11.54 mV
Hist hits 9.84410 Mhits
Hist peak 253.883 khits
1 - 16
17 - 33
34 - 66
67 - 133
134 - 267
268 - 534
535 - 1068

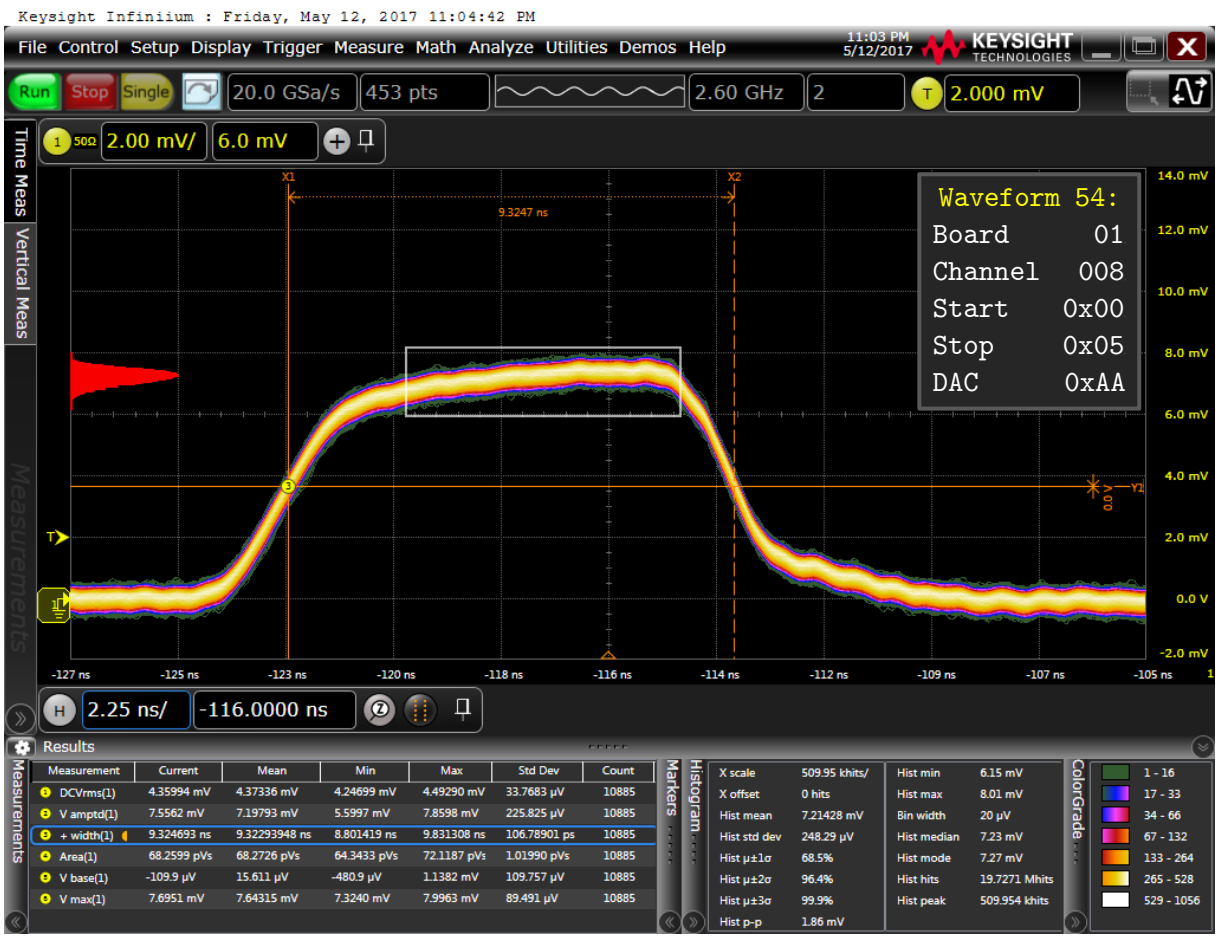
Keysight Infiniium : Friday, May 12, 2017 10:58:38 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:57 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
8.7100 ns
Waveform 49:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0xFF
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 2.91402 mV 2.96294 mV 2.83218 mV 3.08918 mV 33.3720 µV 10131
V amptd(1) 5.0937 mV 4.91033 mV 3.5757 mV 5.5997 mV 205.638 µV 10131
+ width(1) 8.710024 ns 8.97081019 ns 8.449105 ns 9.477845 ns 146.87388 ps 10131
Area(1) 43.6347 pVs 45.1142 pVs 41.3433 pVs 49.8100 pVs 1.03266 pVs 10131
V base(1) -42.4 µV 25.353 µV -548.4 µV 598.5 µV 106.101 µV 10131
V max(1) 5.2112 mV 5.36409 mV 5.0550 mV 5.8714 mV 88.698 µV 10131
X scale 503.15 khits/
X offset 0 hits
Hist mean 4.84831 mV
Hist std dev 309.70 µV
Hist µ±1σ 67.7%
Hist µ±2σ 96.4%
Hist µ±3σ 99.8%
Hist p-p 2.34 mV
Hist min 3.54 mV
Hist max 5.88 mV
Bin width 20 µV
Hist median 4.88 mV
Hist mode 4.96 mV
Hist hits 23.1398 Mhits
Hist peak 503.151 khits
1 - 15
16 - 30
31 - 61
62 - 122
123 - 244
245 - 489
490 - 979

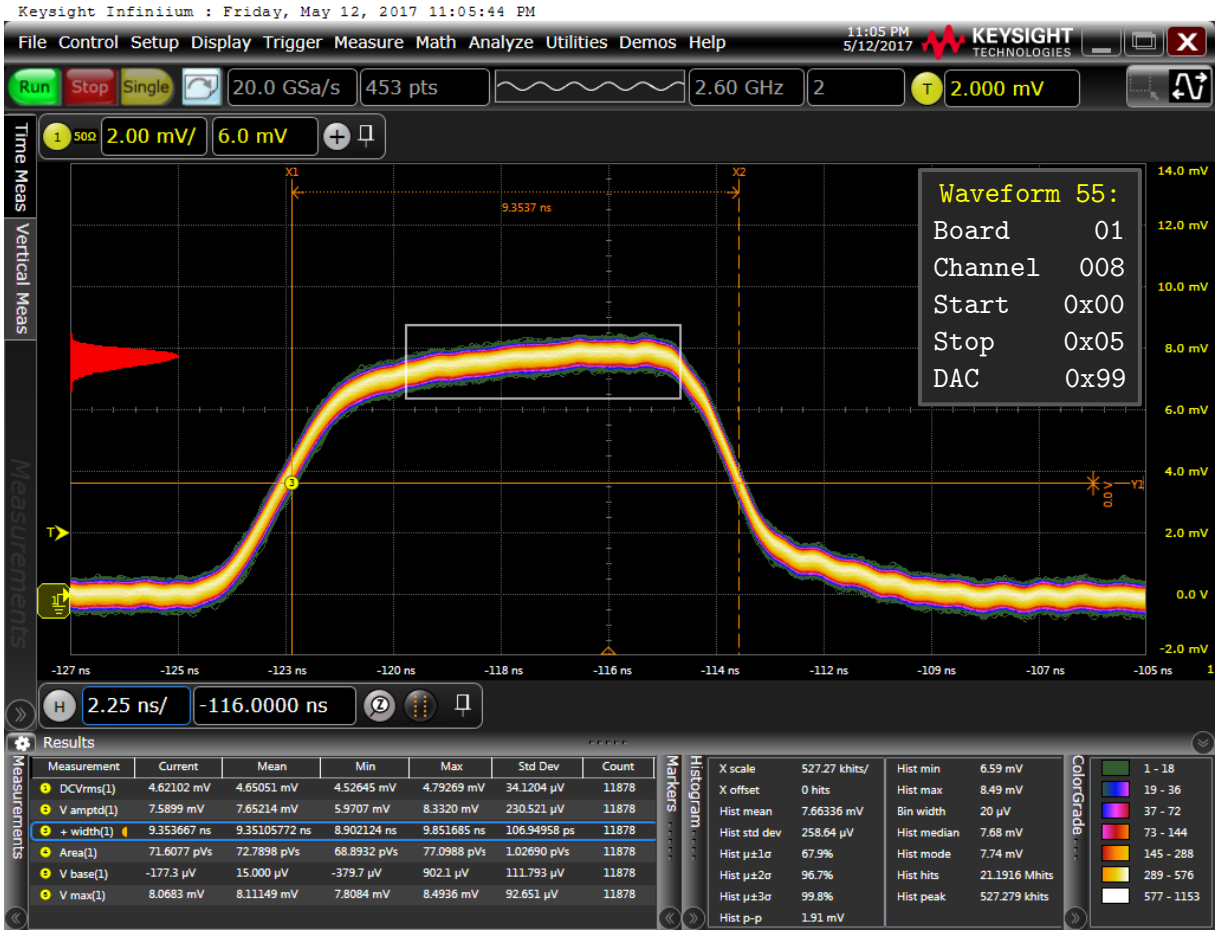

Keysight Infiniium : Friday, May 12, 2017 11:00:16 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
10:59 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.1818 ns
Waveform 50:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0xEE
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.21970 mV 3.23100 mV 3.09696 mV 3.36032 mV 33.5478 µV 10527
V amptd(1) 5.2623 mV 5.36069 mV 3.8456 mV 6.0382 mV 206.584 µV 10527
+ width(1) 9.181791 ns 8.99508378 ns 8.484756 ns 9.707762 ns 139.50709 ps 10527
Area(1) 49.0267 pVs 49.3502 pVs 45.0764 pVs 53.1102 pVs 1.02914 pVs 10527
V base(1) -8.7 µV 29.091 µV -413.5 µV 902.1 µV 107.136 µV 10527
V max(1) 5.8121 mV 5.81610 mV 5.4840 mV 6.1576 mV 88.200 µV 10527
X scale 509.59 khits/
X offset 0 hits
Hist mean 5.33711 mV
Hist std dev 282.46 µV
Hist µ±1σ 68.6%
Hist µ±2σ 96.8%
Hist µ±3σ 99.9%
Hist p-p 2.00 mV
Hist min 4.16 mV
Hist max 6.16 mV
Bin width 20 µV
Hist median 5.37 mV
Hist mode 5.41 mV
Hist hits 21.9254 Mhits
Hist peak 509.595 khits
1 - 15
16 - 31
32 - 62
63 - 124
125 - 249
250 - 499
500 - 999

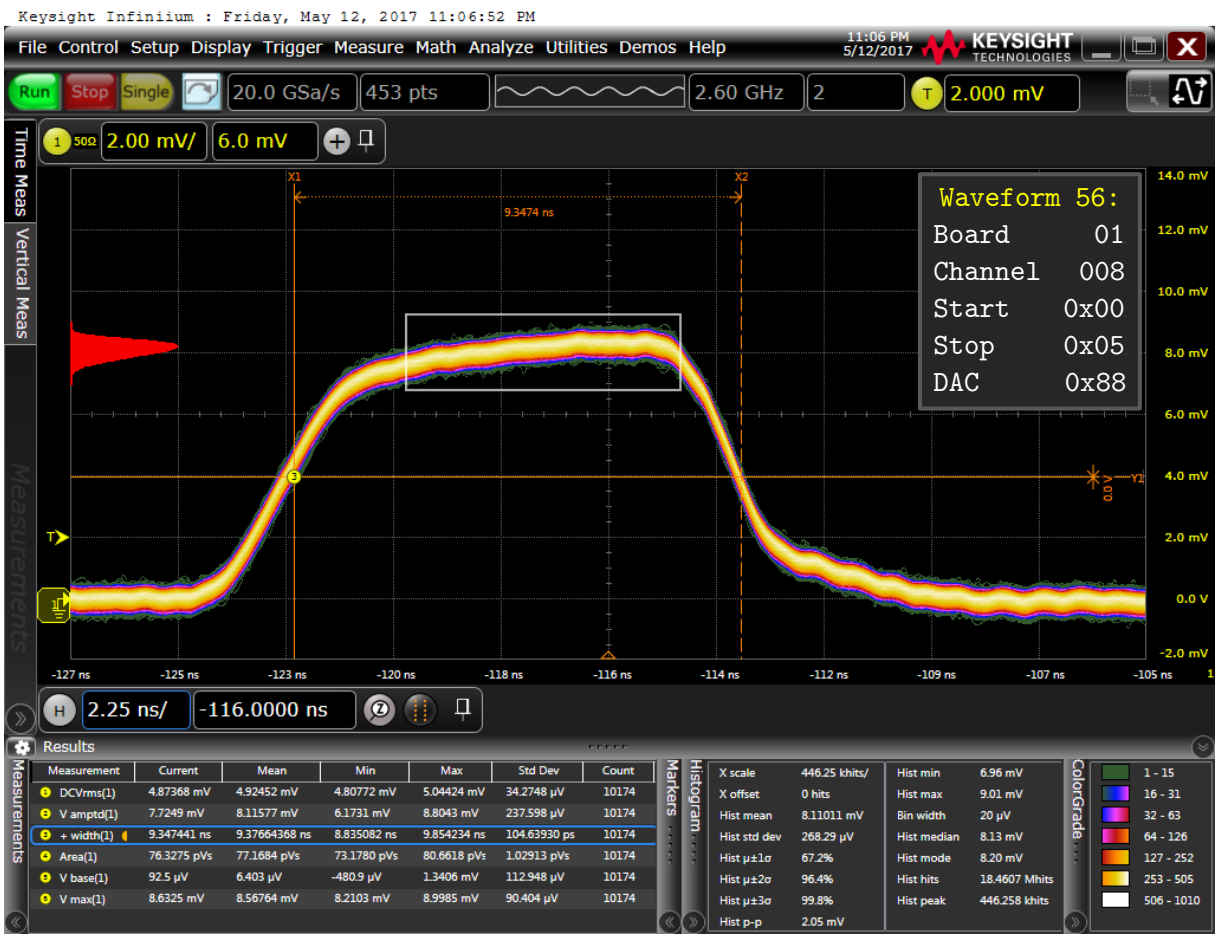
Keysight Infiniium : Friday, May 12, 2017 11:01:20 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:00 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
8.9886 ns
Waveform 51:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0xDD
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.46539 mV 3.51055 mV 3.39276 mV 3.63346 mV 32.9250 µV 10393
V amptd(1) 5.7346 mV 5.81718 mV 4.1829 mV 6.4430 mV 206.946 µV 10393
+ width(1) 8.988647 ns 9.06273128 ns 8.563078 ns 9.634668 ns 131.74240 ps 10393
Area(1) 52.3306 pVs 53.9126 pVs 50.2500 pVs 57.4981 pVs 1.01145 pVs 10393
V base(1) 25.1 µV 28.131 µV -413.5 µV 1.1045 mV 108.524 µV 10393
V max(1) 6.1892 mV 6.27176 mV 5.9665 mV 6.6098 mV 88.917 µV 10393
X scale 502.90 khits/
X offset 0 hits
Hist mean 5.81342 mV
Hist std dev 269.82 µV
Hist µ±1σ 67.5%
Hist µ±2σ 96.3%
Hist µ±3σ 99.8%
Hist p-p 1.97 mV
Hist min 4.65 mV
Hist max 6.62 mV
Bin width 20 µV
Hist median 5.84 mV
Hist mode 5.91 mV
Hist hits 20.5970 Mhits
Hist peak 502.900 khits
1 - 15
16 - 31
32 - 62
63 - 125
126 - 251
252 - 503
504 - 1006

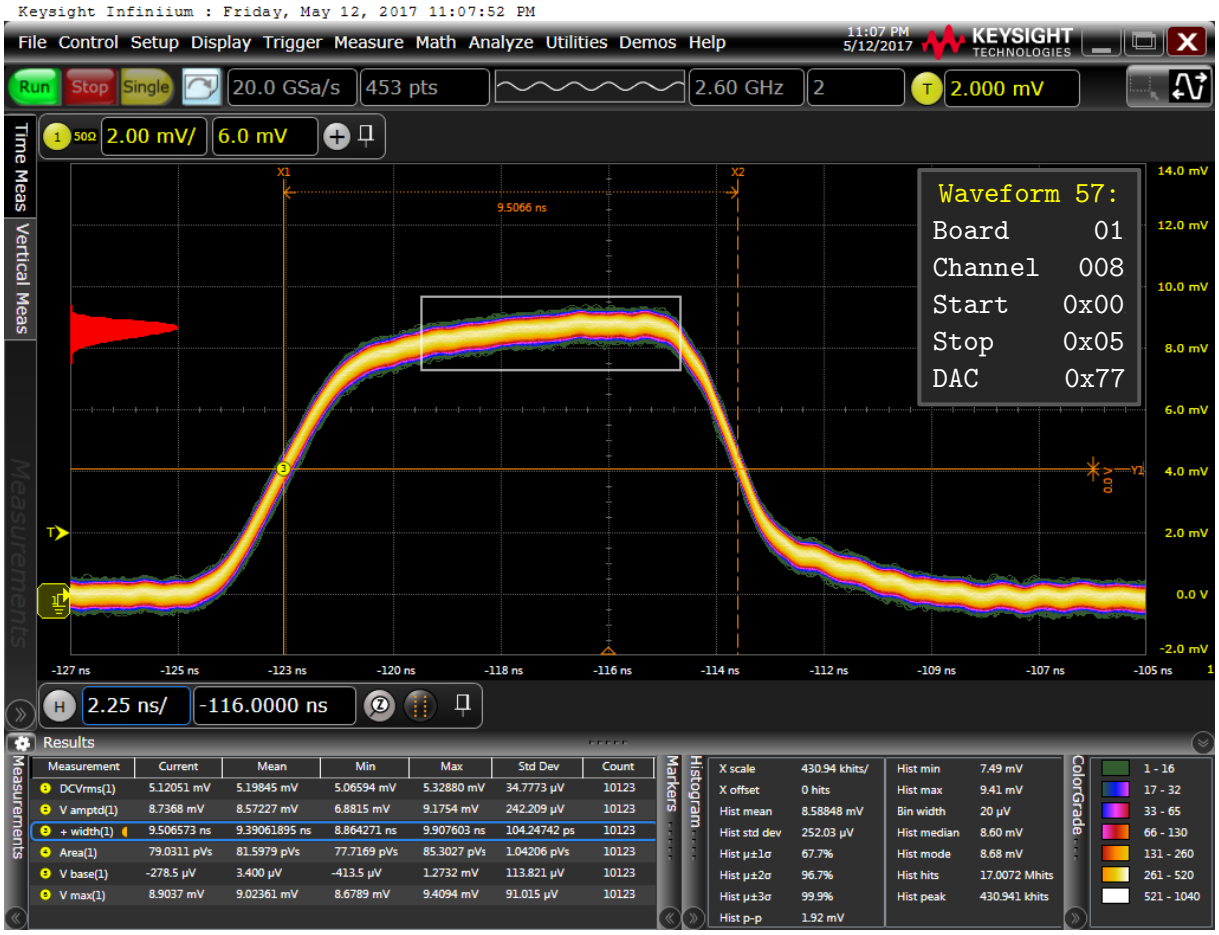

Keysight Infiniium : Friday, May 12, 2017 11:02:18 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:01 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.1295 ns
Waveform 52:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0xCC
14.0 mV
12.0 mV
10.0 mV
8.0 mV
6.0 mV
4.0 mV
2.0 mV
0.0 V
-2.0 mV
-127 ns -125 ns -123 ns -120 ns -118 ns -116 ns -114 ns -112 ns -109 ns -107 ns -105 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 3.84719 mV 3.81586 mV 3.68331 mV 3.96624 mV 33.5633 µV 10330
V amptd(1) 6.1394 mV 6.27235 mV 4.7901 mV 6.9490 mV 216.702 µV 10330
+ width(1) 9.129520 ns 9.23678689 ns 8.753550 ns 9.832091 ns 118.43166 ps 10330
Area(1) 59.9182 pVs 59.1693 pVs 55.1701 pVs 63.9818 pVs 1.02613 pVs 10330
V base(1) 261.2 µV 25.534 µV -447.2 µV 1.0708 mV 111.138 µV 10330
V max(1) 6.7998 mV 6.72982 mV 6.4140 mV 7.1019 mV 88.231 µV 10330
X scale 502.04 khits/
X offset 0 hits
Hist mean 6.26949 mV
Hist std dev 271.71 µV
Hist µ±1σ 68.7%
Hist µ±2σ 96.5%
Hist µ±3σ 99.8%
Hist p-p 1.98 mV
Hist min 5.12 mV
Hist max 7.10 mV
Bin width 20 µV
Hist median 6.29 mV
Hist mode 6.37 mV
Hist hits 20.5591 Mhits
Hist peak 502.047 khits
1 - 15
16 - 30
31 - 61
62 - 123
124 - 247
248 - 494
495 - 988

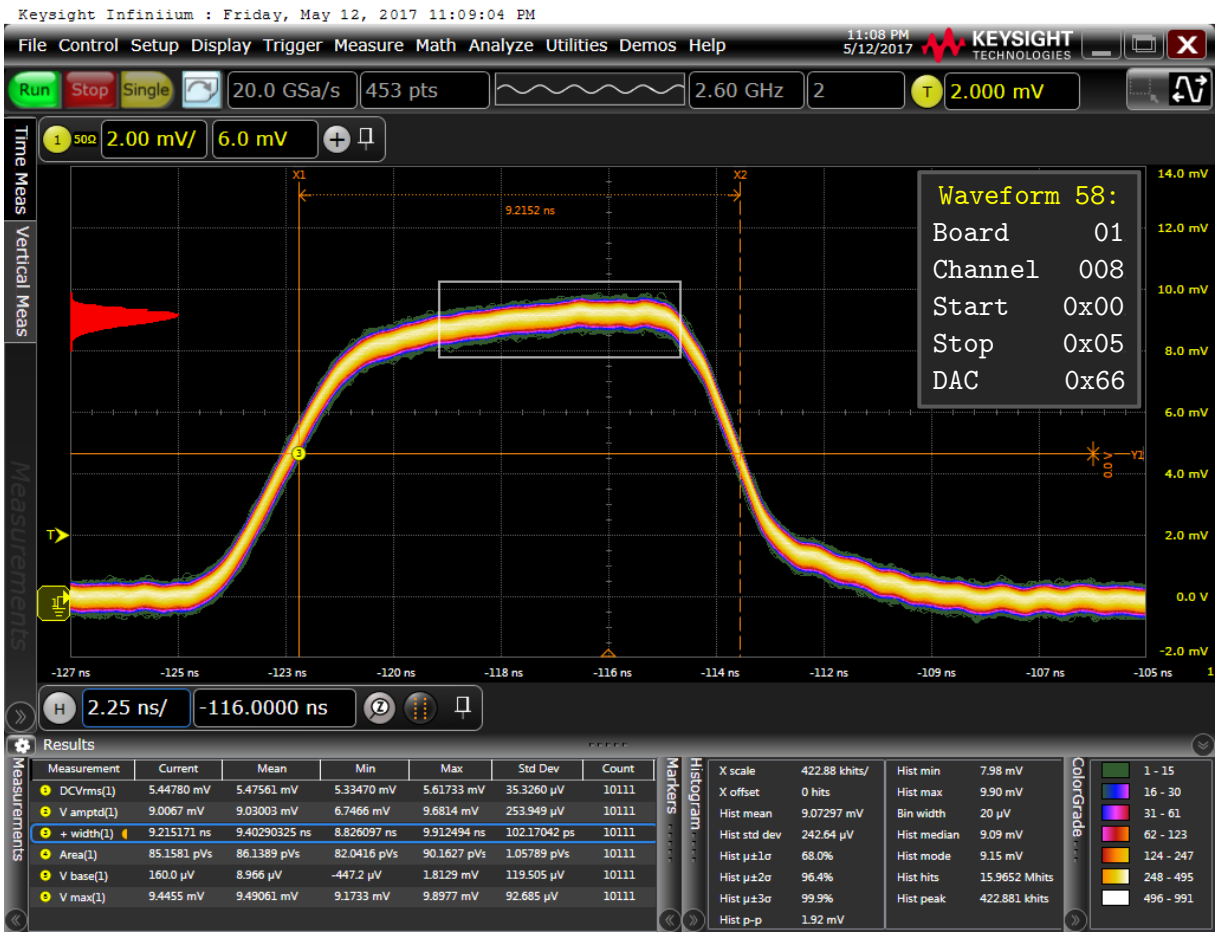
Keysight Infiniium : Friday, May 12, 2017 11:03:41 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:02 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.4366 ns
Waveform 53:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0xBB
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.15088 mV 4.09496 mV 3.96820 mV 4.23197 mV 33.9750 µV 14985
V amptd(1) 6.9153 mV 6.73484 mV 5.1611 mV 7.4213 mV 219.625 µV 14985
+ width(1) 9.436641 ns 9.29299141 ns 8.787891 ns 9.918248 ns 113.10527 ps 14985
Area(1) 64.9583 pVs 63.7101 pVs 59.6243 pVs 68.0281 pVs 1.03009 pVs 14985
V base(1) -8.7 µV 16.357 µV -447.2 µV 1.2394 mV 111.070 µV 14985
V max(1) 7.2958 mV 7.18506 mV 6.8591 mV 7.6998 mV 89.430 µV 14985
X scale 712.60 khits/
X offset 0 hits
Hist mean 6.72661 mV
Hist std dev 270.54 µV
Hist µ±1σ 68.9%
Hist µ±2σ 96.2%
Hist µ±3σ 99.8%
Hist p-p 2.22 mV
Hist min 5.49 mV
Hist max 7.71 mV
Bin width 20 µV
Hist median 6.76 mV
Hist mode 6.82 mV
Hist hits 29.0906 Mhits
Hist peak 712.600 khits
1 - 22
23 - 45
46 - 91
92 - 182
183 - 365
366 - 730
731 - 1461

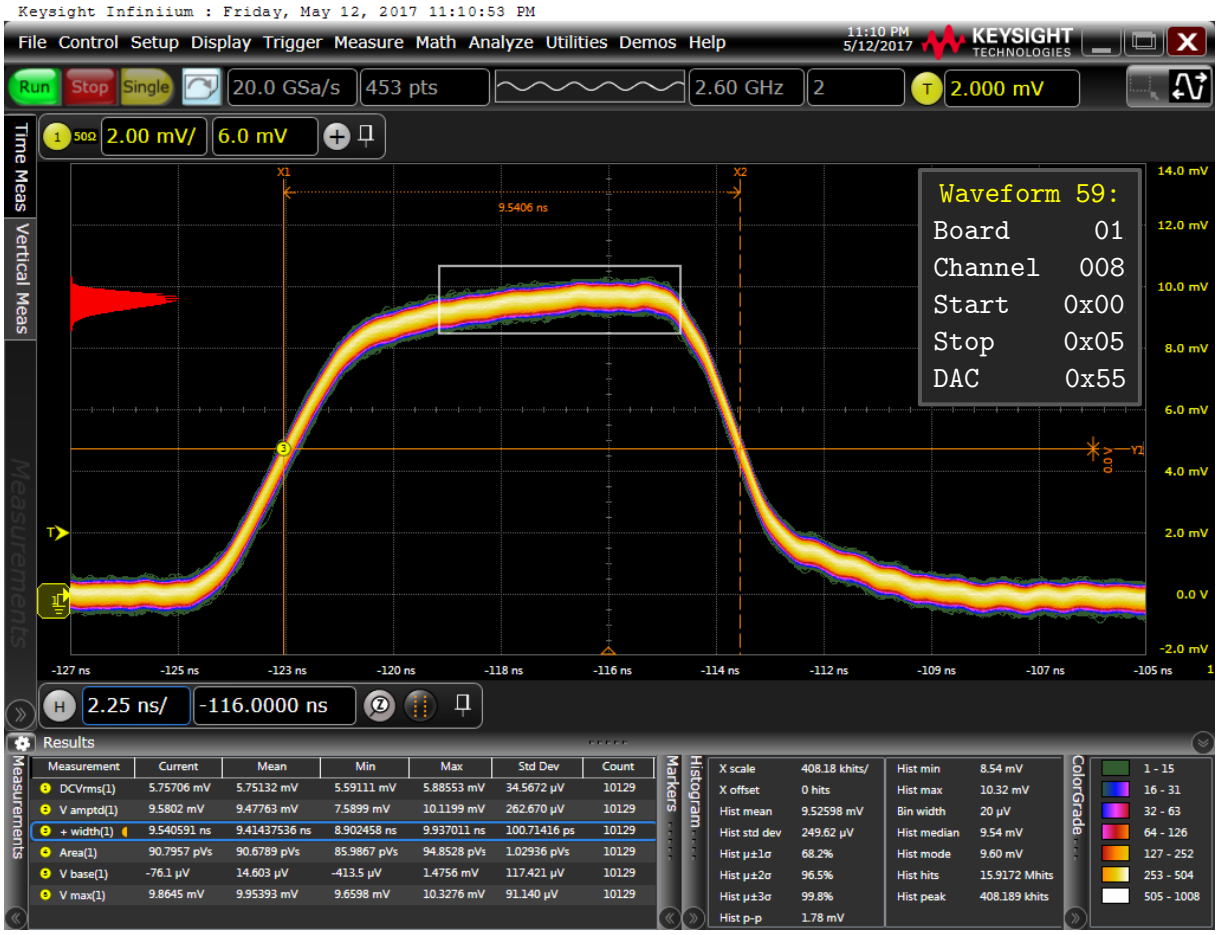

Keysight Infiniium : Friday, May 12, 2017 11:04:42 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:03 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.3247 ns
Waveform 54:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0xAA
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.35994 mV 4.37336 mV 4.24699 mV 4.49290 mV 33.7683 µV 10885
V amptd(1) 7.5562 mV 7.19793 mV 5.5997 mV 7.8598 mV 225.825 µV 10885
+ width(1) 9.324693 ns 9.32293948 ns 8.801419 ns 9.831308 ns 106.78901 ps 10885
Area(1) 68.2599 pVs 68.2726 pVs 64.3433 pVs 72.1187 pVs 1.01990 pVs 10885
V base(1) -109.9 µV 15.611 µV -480.9 µV 1.1382 mV 109.757 µV 10885
V max(1) 7.6951 mV 7.64315 mV 7.3240 mV 7.9963 mV 89.491 µV 10885
X scale 509.95 khits/
X offset 0 hits
Hist mean 7.21428 mV
Hist std dev 248.29 µV
Hist µ±1σ 68.5%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.86 mV
Hist min 6.15 mV
Hist max 8.01 mV
Bin width 20 µV
Hist median 7.23 mV
Hist mode 7.27 mV
Hist hits 19.7271 Mhits
Hist peak 509.954 khits
1 - 16
17 - 33
34 - 66
67 - 132
133 - 264
265 - 528
529 - 1056

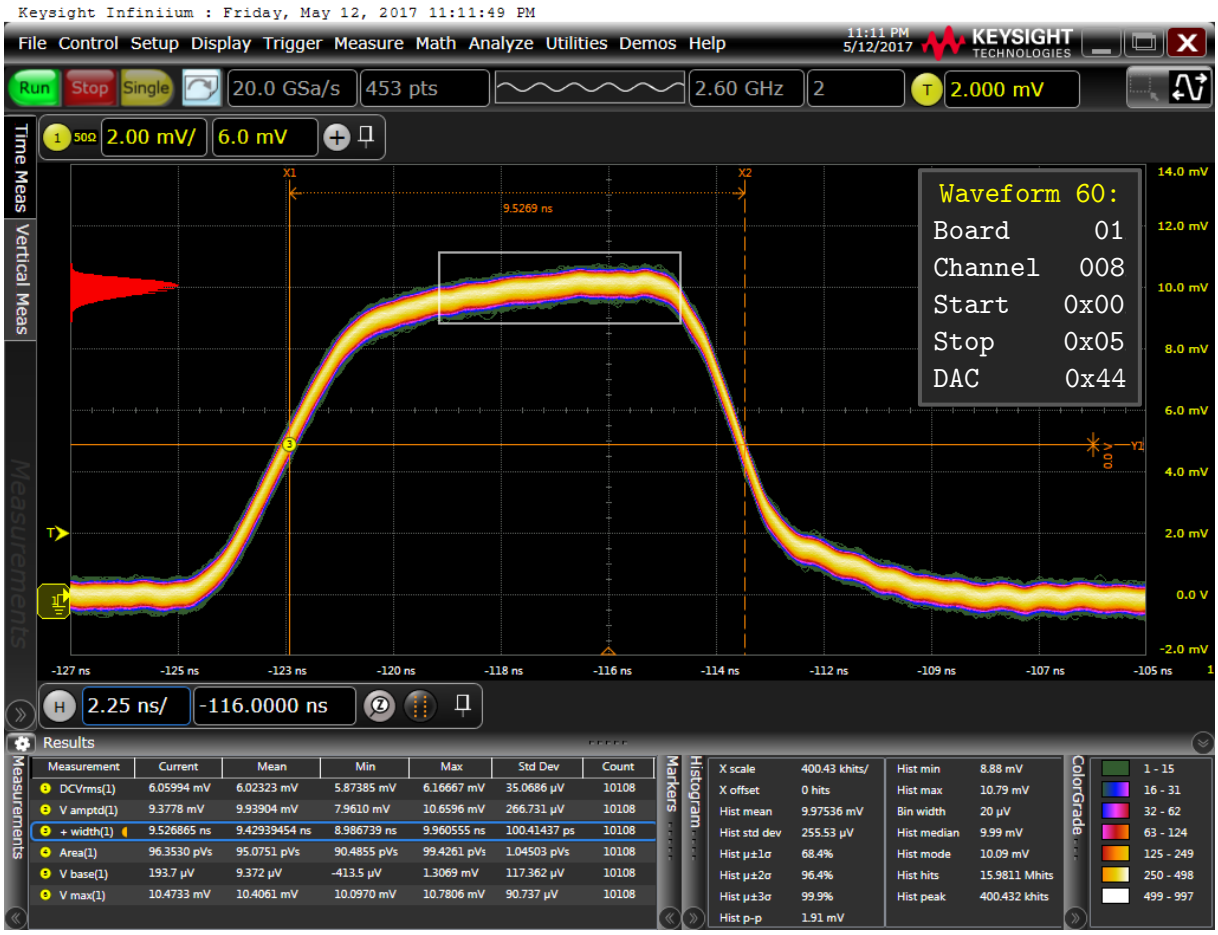

Keysight Infiniium : Friday, May 12, 2017 11:05:44 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:05 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.3537 ns
Waveform 55:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x99
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.62102 mV 4.65051 mV 4.52645 mV 4.79269 mV 34.1204 µV 11878
V amptd(1) 7.5899 mV 7.65214 mV 5.9707 mV 8.3320 mV 230.521 µV 11878
+ width(1) 9.353667 ns 9.35105772 ns 8.902124 ns 9.851685 ns 106.94958 ps 11878
Area(1) 71.6077 pVs 72.7898 pVs 68.8932 pVs 77.0988 pVs 1.02690 pVs 11878
V base(1) -177.3 µV 15.000 µV -379.7 µV 902.1 µV 111.793 µV 11878
V max(1) 8.0683 mV 8.11149 mV 7.8084 mV 8.4936 mV 92.651 µV 11878
Histogram
X scale 527.27 khits/
X offset 0 hits
Hist mean 7.66336 mV
Hist std dev 258.64 µV
Hist µ±1σ 67.9%
Hist µ±2σ 96.7%
Hist µ±3σ 99.8%
Hist p-p 1.91 mV
Hist min 6.59 mV
Hist max 8.49 mV
Bin width 20 µV
Hist median 7.68 mV
Hist mode 7.74 mV
Hist hits 21.1916 Mhits
Hist peak 527.279 khits
ColorGrade
1 - 18
19 - 36
37 - 72
73 - 144
145 - 288
289 - 576
577 - 1153

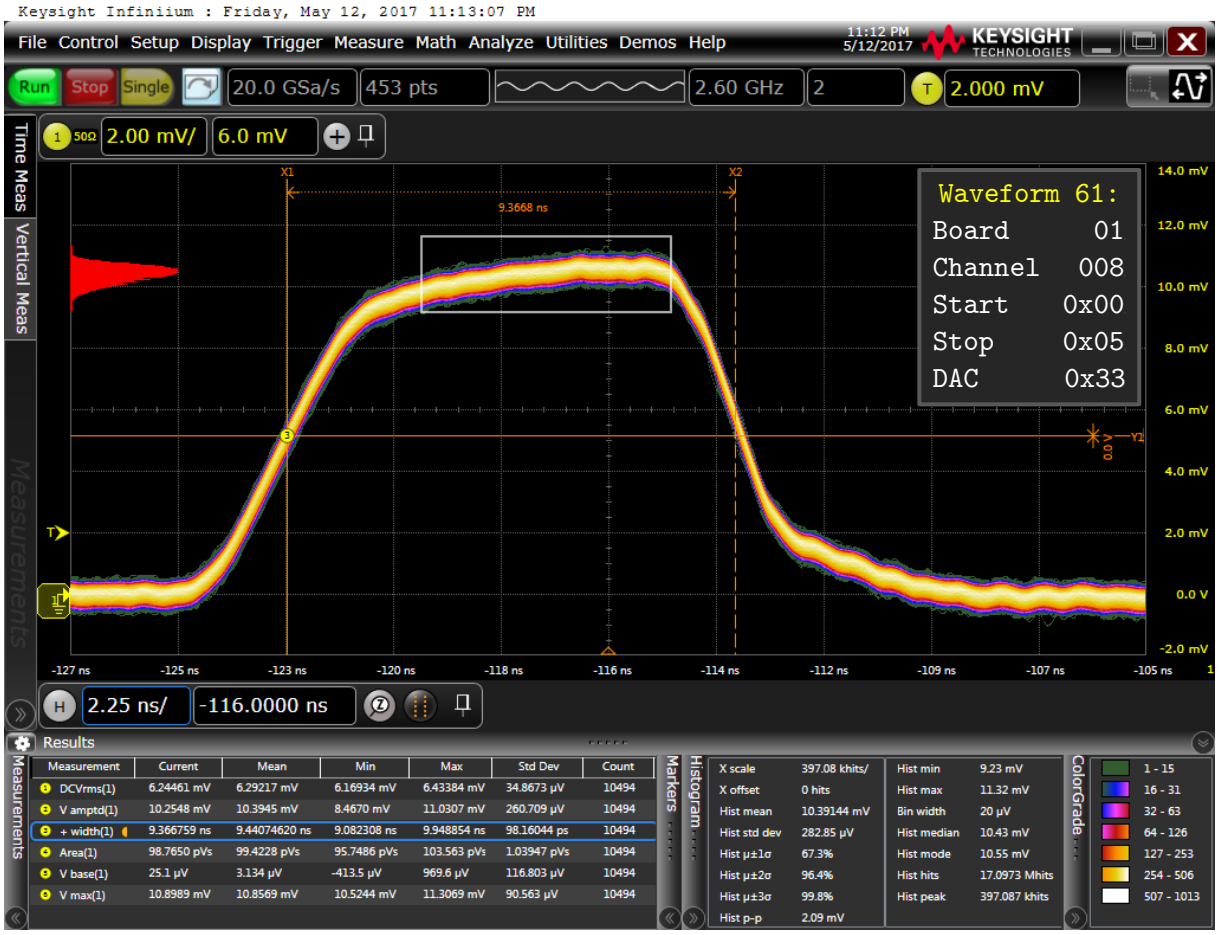

Keysight Infiniium : Friday, May 12, 2017 11:06:52 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:06 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.3474 ns
Waveform 56:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x88
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 4.87368 mV 4.92452 mV 4.80772 mV 5.04424 mV 34.2748 µV 10174
V amptd(1) 7.7249 mV 8.11577 mV 6.1731 mV 8.8043 mV 237.598 µV 10174
+ width(1) 9.347441 ns 9.37664368 ns 8.835082 ns 9.854234 ns 104.63930 ps 10174
Area(1) 76.3275 pVs 77.1684 pVs 73.1780 pVs 80.6618 pVs 1.02913 pVs 10174
V base(1) 92.5 µV 6.403 µV -480.9 µV 1.3406 mV 112.948 µV 10174
V max(1) 8.6325 mV 8.56764 mV 8.2103 mV 8.9985 mV 90.404 µV 10174
Histogram
X scale 446.25 khits/
X offset 0 hits
Hist mean 8.11011 mV
Hist std dev 268.29 µV
Hist µ±1σ 67.2%
Hist µ±2σ 96.4%
Hist µ±3σ 99.8%
Hist p-p 2.05 mV
Hist min 6.96 mV
Hist max 9.01 mV
Bin width 20 µV
Hist median 8.13 mV
Hist mode 8.20 mV
Hist hits 18.4607 Mhits
Hist peak 446.258 khits
ColorGrade
1 - 15
16 - 31
32 - 63
64 - 126
127 - 252
253 - 505
506 - 1010

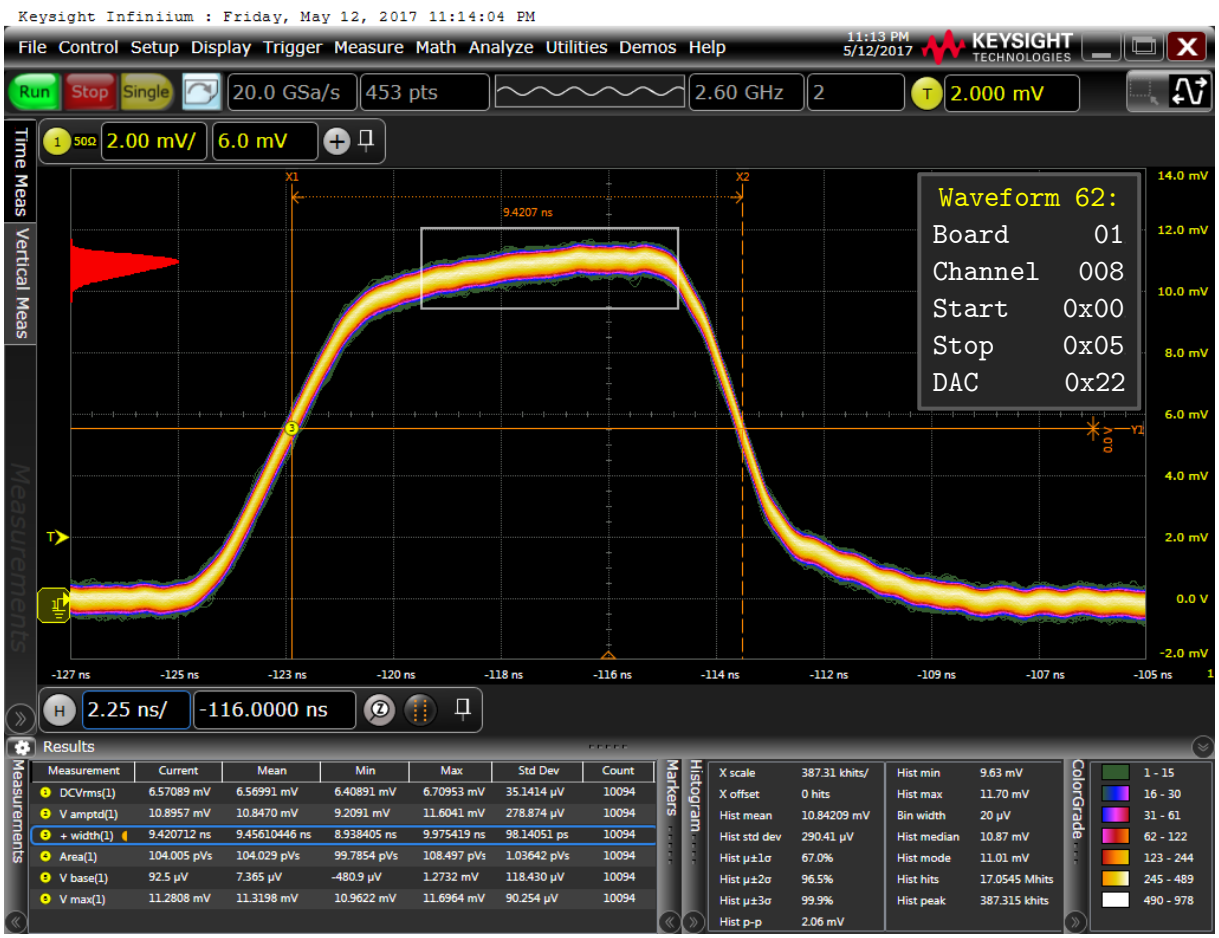

Keysight Infiniium : Friday, May 12, 2017 11:07:52 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:07 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.5066 ns
Waveform 57:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x77
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.12051 mV 5.19845 mV 5.06594 mV 5.32880 mV 34.7773 µV 10123
V amptd(1) 8.7368 mV 8.57227 mV 6.8815 mV 9.1754 mV 242.209 µV 10123
+ width(1) 9.506573 ns 9.39061895 ns 8.864271 ns 9.907603 ns 104.24742 ps 10123
Area(1) 79.0311 pVs 81.5979 pVs 77.7169 pVs 85.3027 pVs 1.04206 pVs 10123
V base(1) -278.5 µV 3.400 µV -413.5 µV 1.2732 mV 113.821 µV 10123
V max(1) 8.9037 mV 9.02361 mV 8.6789 mV 9.4094 mV 91.015 µV 10123
X scale 430.94 khits/
X offset 0 hits
Hist mean 8.58848 mV
Hist std dev 252.03 µV
Hist µ±1σ 67.7%
Hist µ±2σ 96.7%
Hist µ±3σ 99.9%
Hist p-p 1.92 mV
Hist min 7.49 mV
Hist max 9.41 mV
Bin width 20 µV
Hist median 8.60 mV
Hist mode 8.68 mV
Hist hits 17.0072 Mhits
Hist peak 430.941 khits
1 - 16
17 - 32
33 - 65
66 - 130
131 - 260
261 - 520
521 - 1040

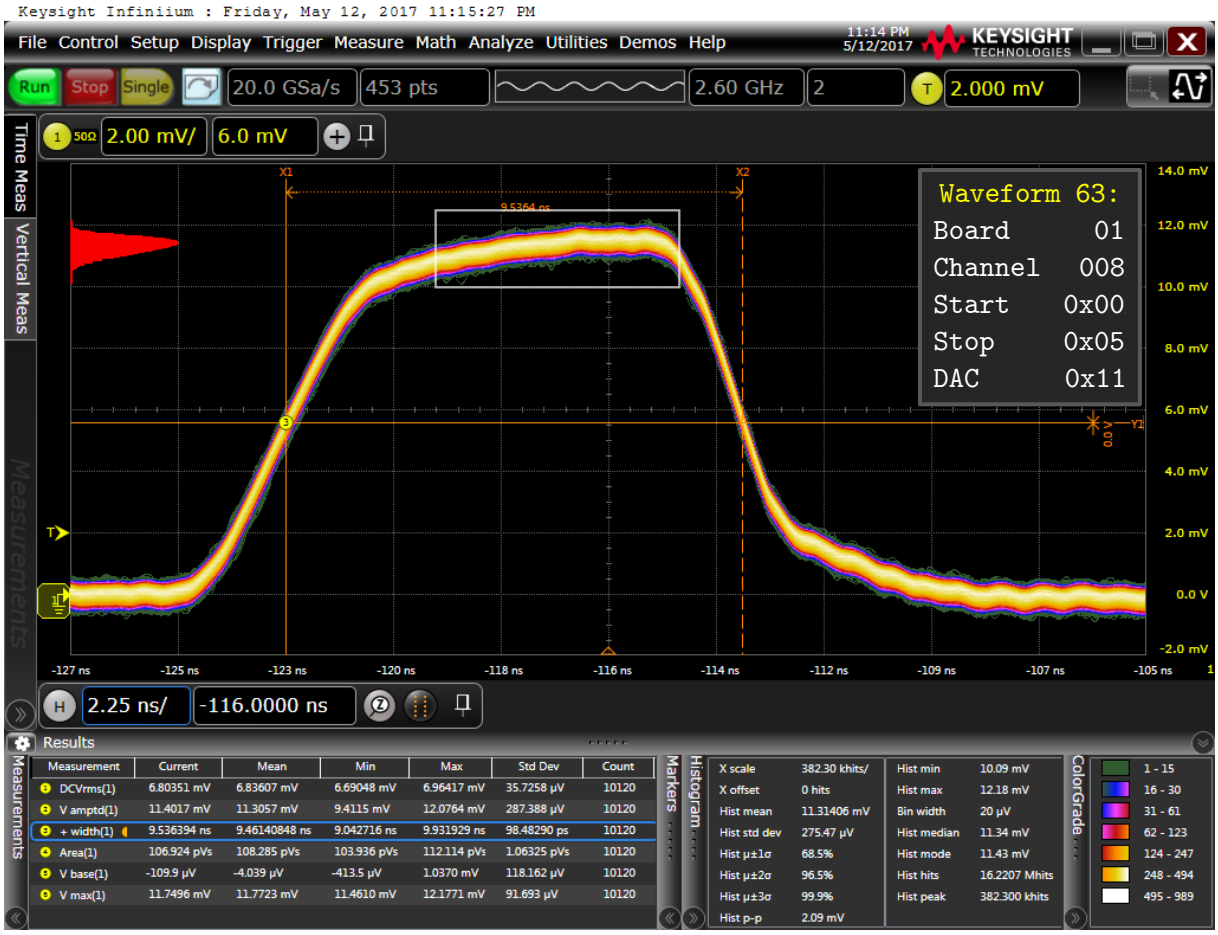

Keysight Infiniium : Friday, May 12, 2017 11:09:04 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:08 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.2152 ns
Waveform 58:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x66
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.44780 mV 5.47561 mV 5.33470 mV 5.61733 mV 35.3260 µV 10111
V amptd(1) 9.0067 mV 9.03003 mV 6.7466 mV 9.6814 mV 253.949 µV 10111
+ width(1) 9.215171 ns 9.40290325 ns 8.826097 ns 9.912494 ns 102.17042 ps 10111
Area(1) 85.1581 pVs 86.1389 pVs 82.0416 pVs 90.1627 pVs 1.05789 pVs 10111
V base(1) 160.0 µV 8.966 µV -447.2 µV 1.8129 mV 119.505 µV 10111
V max(1) 9.4455 mV 9.49061 mV 9.1733 mV 9.8977 mV 92.685 µV 10111
X scale 422.88 khits/
X offset 0 hits
Hist mean 9.07297 mV
Hist std dev 242.64 µV
Hist µ±1σ 68.0%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.92 mV
Hist min 7.98 mV
Hist max 9.90 mV
Bin width 20 µV
Hist median 9.09 mV
Hist mode 9.15 mV
Hist hits 15.9652 Mhits
Hist peak 422.881 khits
1 - 15
16 - 30
31 - 61
62 - 123
124 - 247
248 - 495
496 - 991

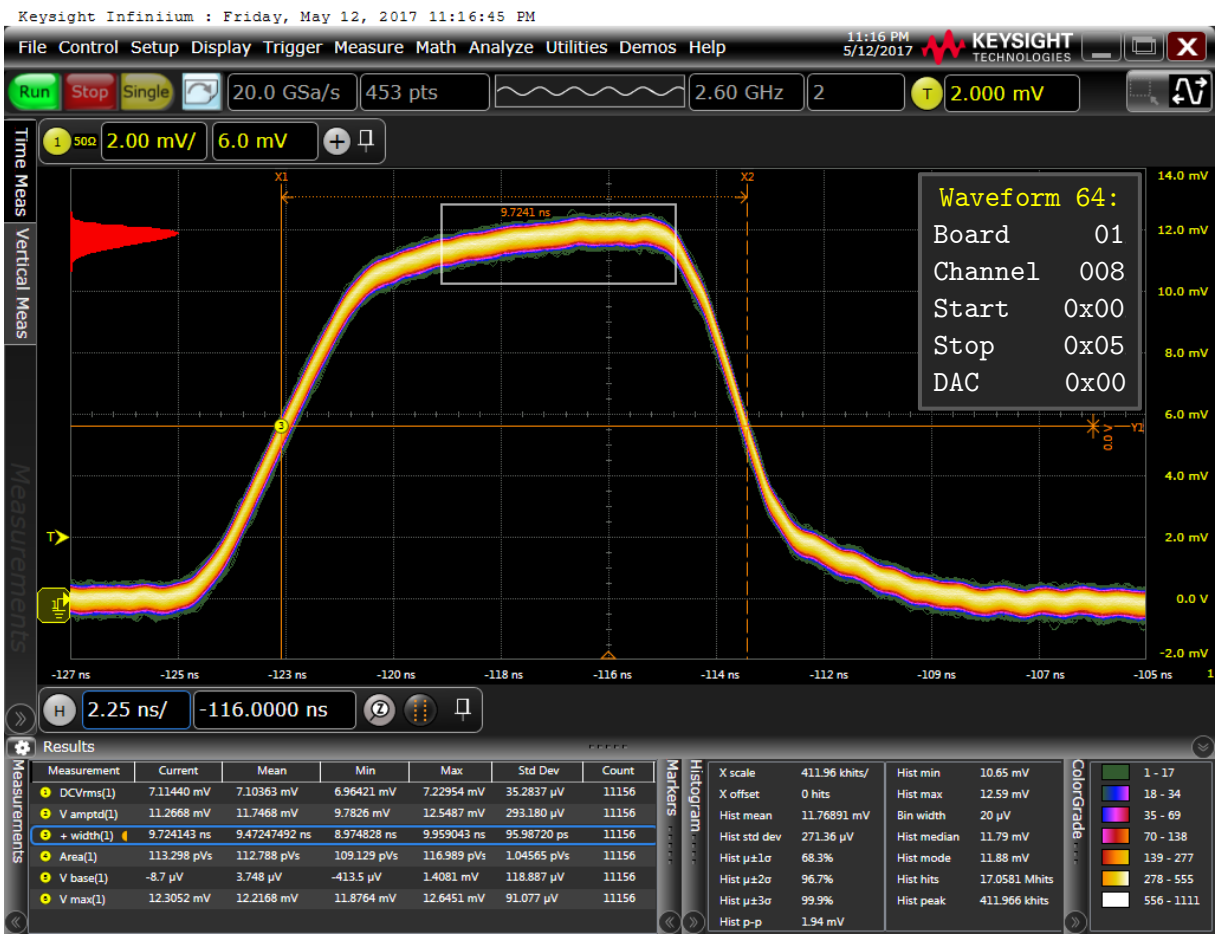

Keysight Infiniium : Friday, May 12, 2017 11:10:53 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:10 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.5406 ns
Waveform 59:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x55
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 5.75706 mV 5.75132 mV 5.59111 mV 5.88553 mV 34.5672 µV 10129
V amptd(1) 9.5802 mV 9.47763 mV 7.5899 mV 10.1199 mV 262.670 µV 10129
+ width(1) 9.540591 ns 9.41437536 ns 8.902458 ns 9.937011 ns 100.71416 ps 10129
Area(1) 90.7957 pVs 90.6789 pVs 85.9867 pVs 94.8528 pVs 1.02936 pVs 10129
V base(1) -76.1 µV 14.603 µV -413.5 µV 1.4756 mV 117.421 µV 10129
V max(1) 9.8645 mV 9.95393 mV 9.6598 mV 10.3276 mV 91.140 µV 10129
X scale 408.18 khits/
X offset 0 hits
Hist mean 9.52598 mV
Hist std dev 249.62 µV
Hist µ±1σ 68.2%
Hist µ±2σ 96.5%
Hist µ±3σ 99.8%
Hist p-p 1.78 mV
Hist min 8.54 mV
Hist max 10.32 mV
Bin width 20 µV
Hist median 9.54 mV
Hist mode 9.60 mV
Hist hits 15.9172 Mhits
Hist peak 408.189 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 252
253 - 504
505 - 1008


Keysight Infiniium : Friday, May 12, 2017 11:11:49 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:11 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.5269 ns
Waveform 60:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x44
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 6.05994 mV 6.02323 mV 5.87385 mV 6.16667 mV 35.0686 µV 10108
V amptd(1) 9.3778 mV 9.93904 mV 7.9610 mV 10.6596 mV 266.731 µV 10108
+ width(1) 9.526865 ns 9.42939454 ns 8.986739 ns 9.960555 ns 100.41437 ps 10108
Area(1) 96.3530 pVs 95.0751 pVs 90.4855 pVs 99.4261 pVs 1.04503 pVs 10108
V base(1) 193.7 µV 9.372 µV -413.5 µV 1.3069 mV 117.362 µV 10108
V max(1) 10.4733 mV 10.4061 mV 10.0970 mV 10.7806 mV 90.737 µV 10108
X scale 400.43 khits/
X offset 0 hits
Hist mean 9.97536 mV
Hist std dev 255.53 µV
Hist µ±1σ 68.4%
Hist µ±2σ 96.4%
Hist µ±3σ 99.9%
Hist p-p 1.91 mV
Hist min 8.88 mV
Hist max 10.79 mV
Bin width 20 µV
Hist median 9.99 mV
Hist mode 10.09 mV
Hist hits 15.9811 Mhits
Hist peak 400.432 khits
1 - 15
16 - 31
32 - 62
63 - 124
125 - 249
250 - 498
499 - 997


Keysight Infiniium : Friday, May 12, 2017 11:13:07 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:12 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.3668 ns
Waveform 61:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x33
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 6.24461 mV 6.29217 mV 6.16934 mV 6.43384 mV 34.8673 µV 10494
V amptd(1) 10.2548 mV 10.3945 mV 8.4670 mV 11.0307 mV 260.709 µV 10494
+ width(1) 9.366759 ns 9.44074620 ns 9.082308 ns 9.948854 ns 98.16044 ps 10494
Area(1) 98.7650 pVs 99.4228 pVs 95.7486 pVs 103.563 pVs 1.03947 pVs 10494
V base(1) 25.1 µV 3.134 µV -413.5 µV 969.6 µV 116.803 µV 10494
V max(1) 10.8989 mV 10.8569 mV 10.5244 mV 11.3069 mV 90.563 µV 10494
X scale 397.08 khits/
X offset 0 hits
Hist mean 10.39144 mV
Hist std dev 282.85 µV
Hist µ±1σ 67.3%
Hist µ±2σ 96.4%
Hist µ±3σ 99.8%
Hist p-p 2.09 mV
Hist min 9.23 mV
Hist max 11.32 mV
Bin width 20 µV
Hist median 10.43 mV
Hist mode 10.55 mV
Hist hits 17.0973 Mhits
Hist peak 397.087 khits
1 - 15
16 - 31
32 - 63
64 - 126
127 - 253
254 - 506
507 - 1013


Keysight Infiniium : Friday, May 12, 2017 11:14:04 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:13 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
9.4207 ns
Waveform 62:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x22
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 6.57089 mV 6.56991 mV 6.40891 mV 6.70953 mV 35.1414 µV 10094
V amptd(1) 10.8957 mV 10.8470 mV 9.2091 mV 11.6041 mV 278.874 µV 10094
+ width(1) 9.420712 ns 9.45610446 ns 8.938405 ns 9.975419 ns 98.14051 ps 10094
Area(1) 104.005 pVs 104.029 pVs 99.7854 pVs 108.497 pVs 1.03642 pVs 10094
V base(1) 92.5 µV 7.365 µV -480.9 µV 1.2732 mV 118.430 µV 10094
V max(1) 11.2808 mV 11.3198 mV 10.9622 mV 11.6964 mV 90.254 µV 10094
X scale 387.31 khits/
X offset 0 hits
Hist mean 10.84209 mV
Hist std dev 290.41 µV
Hist µ±1σ 67.0%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 2.06 mV
Hist min 9.63 mV
Hist max 11.70 mV
Bin width 20 µV
Hist median 10.87 mV
Hist mode 11.01 mV
Hist hits 17.0545 Mhits
Hist peak 387.315 khits
1 - 15
16 - 30
31 - 61
62 - 122
123 - 244
245 - 489
490 - 978


Keysight Infiniium : Friday, May 12, 2017 11:15:27 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:14 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
Waveform 63:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x11
9.5364 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 6.80351 mV 6.83607 mV 6.69048 mV 6.96417 mV 35.7258 µV 10120
V amptd(1) 11.4017 mV 11.3057 mV 9.4115 mV 12.0764 mV 287.388 µV 10120
+ width(1) 9.536394 ns 9.46140848 ns 9.042716 ns 9.931929 ns 98.48290 ps 10120
Area(1) 106.924 pVs 108.285 pVs 103.936 pVs 112.114 pVs 1.06325 pVs 10120
V base(1) -109.9 µV -4.039 µV -413.5 µV 1.0370 mV 118.162 µV 10120
V max(1) 11.7496 mV 11.7723 mV 11.4610 mV 12.1771 mV 91.693 µV 10120
X scale 382.30 khits/
X offset 0 hits
Hist mean 11.31406 mV
Hist std dev 275.47 µV
Hist µ±1σ 68.5%
Hist µ±2σ 96.5%
Hist µ±3σ 99.9%
Hist p-p 2.09 mV
Hist min 10.09 mV
Hist max 12.18 mV
Bin width 20 µV
Hist median 11.34 mV
Hist mode 11.43 mV
Hist hits 16.2207 Mhits
Hist peak 382.300 khits
1 - 15
16 - 30
31 - 61
62 - 123
124 - 247
248 - 494
495 - 989


Keysight Infiniium : Friday, May 12, 2017 11:16:45 PM
File Control Setup Display Trigger Measure Math Analyze Utilities Demos Help
11:16 PM 5/12/2017
KEYSIGHT TECHNOLOGIES
Run Stop Single
20.0 GSa/s
453 pts
2.60 GHz
2
2.000 mV
2.00 mV/
6.0 mV
Waveform 64:
Board 01
Channel 008
Start 0x00
Stop 0x05
DAC 0x00
9.7241 ns
2.25 ns/
-116.0000 ns
Results
Measurement Current Mean Min Max Std Dev Count
DCVrms(1) 7.11440 mV 7.10363 mV 6.96421 mV 7.22954 mV 35.2837 µV 11156
V amptd(1) 11.2668 mV 11.7468 mV 9.7826 mV 12.5487 mV 293.180 µV 11156
+ width(1) 9.724143 ns 9.47247492 ns 8.974828 ns 9.959043 ns 95.98720 ps 11156
Area(1) 113.298 pVs 112.788 pVs 109.129 pVs 116.989 pVs 1.04565 pVs 11156
V base(1) -8.7 µV 3.748 µV -413.5 µV 1.4081 mV 118.887 µV 11156
V max(1) 12.3052 mV 12.2168 mV 11.8764 mV 12.6451 mV 91.077 µV 11156
X scale 411.96 khits/
X offset 0 hits
Hist mean 11.76891 mV
Hist std dev 271.36 µV
Hist µ±1σ 68.3%
Hist µ±2σ 96.7%
Hist µ±3σ 99.9%
Hist p-p 1.94 mV
Hist min 10.65 mV
Hist max 12.59 mV
Bin width 20 µV
Hist median 11.79 mV
Hist mode 11.88 mV
Hist hits 17.0581 Mhits
Hist peak 411.966 khits
1 - 17
18 - 34
35 - 69
70 - 138
139 - 277
278 - 555
556 - 1111

Apêndice G

Dados Tabulados da Seção 4.1.2

Índice de Tabelas deste Apêndice:

Tabela 1 – Largura do pulso em função de diferentes parâmetros 264

Tabela 2 – Área abaixo do pulso em função de diferentes parâmetros 264

Tabela 3 – Carga do pulso em função de diferentes parâmetros 265

Tabela 4 – Tensão de pico do pulso em função de diferentes parâmetros 265

Tabela 1 – Largura do pulso em função de diferentes parâmetros de amplitude e largura.

	FWHM - Full width at half maximum [ns]							
DAC	Width 0x02	Std. Dev.	Width 0x03	Std. Dev.	Width 0x04	Std. Dev.	Width 0x05	Std. Dev.
0xFF	2.078	0.225	4.167	0.124	6.580	0.112	8.971	0.147
0xEE	2.126	0.208	4.190	0.121	6.602	0.111	8.995	0.140
0xDD	2.204	0.195	4.266	0.119	6.683	0.107	9.063	0.132
0xCC	2.366	0.170	4.442	0.122	6.851	0.116	9.237	0.118
0xBB	2.446	0.157	4.501	0.119	6.900	0.115	9.293	0.113
0xAA	2.516	0.156	4.544	0.122	6.948	0.118	9.323	0.107
0x99	2.561	0.153	4.573	0.120	6.967	0.115	9.351	0.107
0x88	2.612	0.146	4.599	0.118	6.987	0.115	9.377	0.105
0x77	2.641	0.148	4.622	0.117	7.003	0.114	9.391	0.104
0x66	2.679	0.144	4.641	0.115	7.018	0.113	9.403	0.102
0x55	2.721	0.137	4.658	0.109	7.033	0.112	9.414	0.101
0x44	2.749	0.136	4.680	0.110	7.040	0.108	9.429	0.100
0x33	2.784	0.137	4.699	0.109	7.055	0.105	9.441	0.098
0x22	2.821	0.134	4.718	0.105	7.061	0.105	9.456	0.098
0x11	2.859	0.133	4.731	0.103	7.076	0.107	9.461	0.098
0x00	2.892	0.129	4.751	0.101	7.092	0.114	9.472	0.096

Tabela 2 – Área abaixo do pulso em função de diferentes parâmetros de amplitude e largura.

	Area [pVs]							
DAC	Width 0x02	Std. Dev.	Width 0x03	Std. Dev.	Width 0x04	Std. Dev.	Width 0x05	Std. Dev.
0xFF	5.153	1.070	19.461	1.057	32.388	0.987	45.114	1.033
0xEE	5.701	1.044	21.525	1.048	35.504	1.000	49.350	1.029
0xDD	6.576	1.073	23.708	1.042	38.951	1.014	53.913	1.011
0xCC	8.447	1.053	26.672	1.047	42.907	1.019	59.169	1.026
0xBB	9.671	1.054	28.769	1.046	46.538	1.011	63.710	1.030
0xAA	10.564	1.073	31.090	1.055	50.038	1.011	68.273	1.020
0x99	11.468	1.093	33.371	1.040	53.270	1.017	72.790	1.027
0x88	12.246	1.105	35.489	1.048	56.521	1.033	77.168	1.029
0x77	12.904	1.079	37.623	1.067	59.590	1.034	81.598	1.042
0x66	13.695	1.081	39.798	1.023	62.974	1.020	86.139	1.058
0x55	14.597	1.097	41.085	1.027	66.139	1.054	90.679	1.029
0x44	15.340	1.084	44.126	1.009	69.509	1.048	95.075	1.045
0x33	16.072	1.102	46.259	1.019	72.708	1.032	99.423	1.039
0x22	17.156	1.103	48.652	1.030	75.905	1.055	104.029	1.036
0x11	18.077	1.110	50.837	1.008	79.280	1.051	108.285	1.063
0x00	18.949	1.144	53.182	1.014	82.631	1.053	112.788	1.046

Tabela 3 – Carga do pulso em função de diferentes parâmetros de amplitude e largura.

	Calculated Charge @ 50 Ohm [pC]							
DAC	Width 0x02	Std. Dev.	Width 0x03	Std. Dev.	Width 0x04	Std. Dev.	Width 0x05	Std. Dev.
0xFF	0.10307	0.02140	0.38921	0.02114	0.64775	0.01975	0.90228	0.02065
0xEE	0.11402	0.02088	0.43050	0.02095	0.71008	0.02000	0.98700	0.02058
0xDD	0.13152	0.02146	0.47417	0.02085	0.77902	0.02029	1.07825	0.02023
0xCC	0.16893	0.02107	0.53344	0.02094	0.85813	0.02037	1.18339	0.02052
0xBB	0.19341	0.02108	0.57537	0.02092	0.93076	0.02021	1.27420	0.02060
0xAA	0.21127	0.02146	0.62180	0.02109	1.00076	0.02023	1.36545	0.02040
0x99	0.22936	0.02185	0.66741	0.02079	1.06539	0.02035	1.45580	0.02054
0x88	0.24492	0.02210	0.70977	0.02097	1.13042	0.02066	1.54337	0.02058
0x77	0.25808	0.02158	0.75246	0.02134	1.19181	0.02068	1.63196	0.02084
0x66	0.27390	0.02162	0.79596	0.02047	1.25948	0.02041	1.72278	0.02116
0x55	0.29195	0.02194	0.82170	0.02054	1.32278	0.02108	1.81358	0.02059
0x44	0.30680	0.02168	0.88253	0.02019	1.39018	0.02096	1.90150	0.02090
0x33	0.32144	0.02203	0.92518	0.02038	1.45416	0.02063	1.98846	0.02079
0x22	0.34312	0.02207	0.97304	0.02059	1.51809	0.02111	2.08058	0.02073
0x11	0.36154	0.02220	1.01674	0.02016	1.58559	0.02102	2.16570	0.02127
0x00	0.37898	0.02288	1.06364	0.02029	1.65263	0.02106	2.25576	0.02091

Tabela 4 – Tensão de pico do pulso em função de diferentes parâmetros de amplitude e largura.

	Positive Peak [mV]							
DAC	Width 0x02	Std. Dev.	Width 0x03	Std. Dev.	Width 0x04	Std. Dev.	Width 0x05	Std. Dev.
0xFF	2.557	0.157	4.770	0.105	5.175	0.097	5.364	0.089
0xEE	2.744	0.159	5.182	0.108	5.618	0.092	5.816	0.088
0xDD	3.049	0.156	5.604	0.108	6.066	0.098	6.272	0.089
0xCC	3.592	0.153	6.067	0.108	6.522	0.099	6.730	0.088
0xBB	3.901	0.159	6.481	0.110	6.977	0.100	7.185	0.089
0xAA	4.157	0.161	6.916	0.111	7.433	0.100	7.643	0.089
0x99	4.380	0.160	7.337	0.113	7.876	0.102	8.111	0.093
0x88	4.606	0.166	7.752	0.115	8.333	0.102	8.568	0.090
0x77	4.790	0.161	8.173	0.117	8.771	0.102	9.024	0.091
0x66	4.992	0.166	8.591	0.116	9.228	0.104	9.491	0.093
0x55	5.216	0.167	8.996	0.117	9.670	0.106	9.954	0.091
0x44	5.403	0.166	9.419	0.117	10.120	0.106	10.406	0.091
0x33	5.605	0.165	9.831	0.118	10.557	0.104	10.857	0.091
0x22	5.828	0.167	10.252	0.116	11.004	0.106	11.320	0.090
0x11	6.058	0.170	10.664	0.120	11.454	0.109	11.772	0.092
0x00	6.276	0.169	11.067	0.118	11.892	0.111	12.217	0.091

Apêndice H

Código C^{++} da Seção 4.2.2

```
0000 /*
0001   Este código foi desenvolvido no âmbito do projeto SciFi (LHCb) para implementar um
0002   algoritmo de testes automatizado para caracterização do Disp. Injetor de Carga.
0003
0004   O algoritmo de testes envolve o comando sincronizado do Disp. Injetor de Carga e de
0005   um osciloscópio, que atua como aquisitor de dados durante os testes. Uma série de
0006   pulsos idênticos é injetada no osciloscópio, que realiza diversas medidas estatís-
0007   ticas para cada série de pulsos. Comandos são então enviados para que o osciloscópio
0008   páre a aquisição, grave em disco alguns arquivos selecionados e fique em espera. A
0009   seguir,  comandos são enviados ao Dispositivo Injetor de Carga, para que se alterem
0010   alguns parâmetros selecionados dos pulsos, de modo que se crie um padrão de teste.
0011   O osciloscópio é então comandado a reiniciar a aquisição durante algum tempo até que
0012   este ciclo se complete e reinicie.
0013
0014   Neste código foram incluídos os procedimentos para controle remoto do Osciloscópio
0015   Agilent Infiniium do LAFEX, baseados na emulação de um teclado e no pressionamento
0016   virtual da sequência de teclas apropriada. Este método foi testado na unidade de
0017   S/N MY48240306 e não há garantia alguma de que vá funcionar em outra unidade!
0018
0019   Devido à complexidade do algoritmo sugere-se que o hardware Arduino DUE seja uti-
0020   lizado, por contar - simultâneamente - com porta USB de emulação de teclado (para
0021   conexão ao osciloscópio), porta USB de programação (para visualização de mensagens
0022   e execução de comandos de usuário em terminal serial USB), porta I2C para controle
0023   do Injetor e por este hardware operar em 3v3, tensão nativa do Injetor, dispensan-
0024   do assim o uso de um level shifter I2C.
0025 */
0026
0027 #include "Keyboard.h"
0028 #include "Mouse.h"
0029 #include <Wire.h>
0030
0031 int value;
0032 byte received;
0033 byte state;
0034 byte test_state, TestAlgorithm_state;
0035 byte SlaveAddress = 0x00;
0036 byte next_write;
0037 byte UARTbufferIndex;
0038 byte TestChannel;
0039 byte AlgorithmAmplitude, AlgorithmAmplitudeStep, AlgorithmDacReg;
0040 byte AlgorithmWidth;
0041 byte AlgorithmChannel;
0042 byte AlgTake1_Channel;
0043 byte AlgTake2_Channel;
0044 byte AlgTake3_Channel;
0045 byte AlgTake4_Channel;
0046 byte AlgorithmChannelCount;
0047 byte AlgorithmMaskData;
0048 unsigned int AlgorithmMaskRegA;
0049 unsigned int testStep, numSteps;
0050 unsigned int errorCounter;
0051 boolean goState, goTestAlgorithm, endState, AbortTestAlgorithm, XTalkRUN;
0052 unsigned int TxSeq = 0x0000;
0053 boolean halfByte=true;
0054 boolean illegalInput=false;
0055 unsigned int InitialAddr, NumberAddr, BufferLength;
0056 unsigned int TestNumber;
0057 boolean TxPaused, TxLastState = false;
0058 byte CRCtx, CRCrx;
0059 unsigned long start_timestamp;
0060 unsigned long start_time, start_cents, start_second, start_minute, start_hour;
0061 unsigned long volatile_timestamp;
0062 char TestIdentifier[29], XTalkID;
0063 unsigned long acq_counter;
0064
0065 unsigned long DURATION_OF_AQUISITION = 60000; // Data aquisition period in milisseconds
0066 unsigned int responseDelay = 500;             // Delay between keyboard commands
0067
0068 // set pin numbers for the button:
0069 const int goButton = 2;
0070
0071
```

```
0072 void setup() {
0073   // initialize variables:
0074   goState = false;
0075   // initialize Hardware (I2C, Mouse, Keyboard and Serial):
0076   Wire.begin();
0077   Mouse.begin();
0078   Keyboard.begin();
0079   Serial.begin(115200);
0080   // Print welcome message at user terminal:
0081   Serial.println();
0082   Serial.println("----------------------------");
0083   Serial.println("- LHCb UPGRADE TEST SYSTEM -");
0084   Serial.println("- INJECTOR TEST CONTROLLER -");
0085   Serial.println("----------------------------");
0086   // Look for I2C slave:
0087   I2Cscan();
0088   // Print command list at user terminal:
0089   CommandList();
0090 }
0091
0092 void loop() {
0093 // Update UART Buffer Index:
0094   UARTbufferIndex = (byte)Serial.available();
0095
0096 // Call for Test Algorithm State Machine subroutine:
0097   TestAlgorithm();
0098
0099 // Call for Test Procedure State Machine subroutine:
0100   TestProcStateMachine();
0101
0102 // Call for User Interface State Machine subroutine:
0103   if(UARTbufferIndex>0) {
0104     UserStateMachine();
0105   }
0106 }
0107
0108 void TestAlgorithm() {
0109   switch (TestAlgorithm_state)
0110   {
0111     case 0x00: // IDLE
0112       if (goTestAlgorithm == true) {
0113       goTestAlgorithm = false;
0114       TestAlgorithm_state = 0x01;
0115       }
0116       break;
0117     case 0x01: // Setup Injector w/ channel 7 on
0118       errorCounter=0;
0119       AlgorithmChannelCount = 0x00;
0120       AlgorithmChannel = AlgTake1_Channel;
0121       AbortTestAlgorithm = false;
0122       Serial.print("TASK - Setting Up Injector: ");
0123       SimpleI2Cwrite(0x0881, 0x27, true);  // Setup Prescaler
0124       if(SimpleI2Cread(0x0881, true)!=0x27){
0125         Serial.println("ERRR - I2C Communication Error !!!");
0126         errorCounter++;
0127       }
0128       TestAlgorithm_state = 0x02;
0129       testStep = 0;
0130       numSteps = ((0xFF/AlgorithmAmplitudeStep)+1)*7*4+4;
0131       break;
0132     case 0x02: //
0133       Serial.print("TASK - Setting On-Off Mask: ");
0134       I2CMaskAllOff();
0135       Serial.print("TASK - Setting On-Off Mask: ");
0136       AlgorithmMaskData = 0x01 << (AlgorithmChannel & 0x07);
0137       AlgorithmMaskRegA = 0x0800 + (AlgorithmChannel >> 3);
0138       SimpleI2Cwrite(AlgorithmMaskRegA, AlgorithmMaskData, true);  // Setup On-Off Mask
0139       if(SimpleI2Cread(AlgorithmMaskRegA, true)!=AlgorithmMaskData){
0140         Serial.println("ERRR - I2C Communication Error !!!");
0141         errorCounter++;
0142       }
0143       TestAlgorithm_state = 0x03;
```

```
0144        break;
0145      case 0x03: // Get XTALK data
0146        Serial.print("INFO - Setting Scope for X-Talk analysis of Input ");
0147        Serial.print(AlgorithmChannelCount+1);
0148        Serial.print(" (channel 0x");
0149        if (AlgorithmChannel < 0x10) Serial.print("0");
0150        Serial.print(AlgorithmChannel);
0151        Serial.println(")");
0152        AlgorithmAmplitude = 0x00;
0153        AlgorithmWidth = 0x06;
0154        loadSetup(0x00, AlgorithmChannelCount+1);
0155        TestAlgorithm_state = 0x05;
0156        XTalkRUN = true;
0157        break;
0158      case 0x04: // Get Channel data
0159        Serial.print("INFO - Setting Scope for Input ");
0160        Serial.print(AlgorithmChannelCount+1);
0161        Serial.print(" (channel 0x");
0162        if (AlgorithmChannel < 0x10) Serial.print("0");
0163        Serial.print(AlgorithmChannel);
0164        Serial.println(")");
0165        AlgorithmAmplitude = 0xFF;
0166        AlgorithmWidth = 0x02;
0167        loadSetup(0x01, AlgorithmChannelCount+1);
0168        TestAlgorithm_state = 0x05;
0169        break;
0170      case 0x05: //
0171        Serial.print("TASK - Setting Pulse Width: ");
0172        SimpleI2Cwrite(0x0200, AlgorithmWidth, true);  // Setup Pulse Width
0173        if(SimpleI2Cread(0x0200, true)!=AlgorithmWidth){
0174          Serial.println("ERRR - I2C Communication Error !!!");
0175          errorCounter++;
0176        }
0177        TestAlgorithm_state = 0x06;
0178        break;
0179      case 0x06: //
0180        debugAquisition();
0181        start_time   = millis();
0182        start_second = start_time / 1000;
0183        start_minute = start_second / 60;
0184        start_hour   = start_minute / 60;
0185        start_cents  = start_time   - start_second*1000;
0186        start_second = start_second - start_minute*60;
0187        start_minute = start_minute - start_hour*60;
0188        testStep++;
0189        Serial.print("INFO - Step ");
0190        Serial.print(testStep);
0191        Serial.print(" of ");
0192        Serial.print(numSteps);
0193        Serial.print(" (");
0194        Serial.print(errorCounter);
0195        Serial.print(" error) Millis:");
0196        Serial.println(start_time);
0197        Serial.print("TASK - Setting Pulse Ampli: ");
0198        AlgorithmDacReg = DACregister(AlgorithmChannel);
0199        // SimpleI2Cwrite(AlgorithmDacReg, AlgorithmAmplitude, true);  // Setup Amplitude
0200        // if(SimpleI2Cread(AlgorithmDacReg, true)!=AlgorithmAmplitude){
0201          // Serial.println("ERRR - I2C Communication Error !!!");
0202          // errorCounter++;
0203        // }
0204
0205        // ATENÇÃO!!!! GAMBIARRA PROVISÓRIA PARA RESOLVER A SENSIBILIDADE DAS AMPLITUDES
0206        // ISTO PRECISA SER CONSERTADO !!!!!!
0207
0208        Serial.println("GAMBIARRA ATIVADA");
0209        for (byte u=0; u <= 0x0F; u++) {
0210          SimpleI2Cwrite(u, AlgorithmAmplitude, false);  // Setup Amplitude
0211          if(SimpleI2Cread(u, false)!=AlgorithmAmplitude){
0212            Serial.println("ERRR - I2C Communication Error !!!");
0213            errorCounter++;
0214          }
0215        }
```

```
0216
0217        XTalkID = 'i';
0218        if (XTalkRUN) XTalkID = 'x';
0219        sprintf (TestIdentifier, "r%04X%c_c%02Xw%02Xa%02Xt%02dh%02dm%02ds%03d", TestNumber,
XTalkID, AlgorithmChannel, AlgorithmWidth, AlgorithmAmplitude, start_hour, start_minute,
start_second, start_cents);
0220        endState=false;
0221        goState = true;
0222        TestAlgorithm_state = 0x07;
0223        break;
0224      case 0x07: //
0225        if(endState) {
0226          endState=false;
0227          if (AbortTestAlgorithm) {
0228            Serial.println("INFO - Algorithm Aborted by user");
0229            TestAlgorithm_state = 0x08;
0230            break;
0231          }
0232          if (XTalkRUN) {
0233            TestAlgorithm_state = 0x04;
0234            XTalkRUN = false;
0235            break;
0236          }
0237          if (AlgorithmAmplitude > AlgorithmAmplitudeStep-1) {
0238            AlgorithmAmplitude = AlgorithmAmplitude - AlgorithmAmplitudeStep;
0239            TestAlgorithm_state = 0x06;
0240          } else {
0241            AlgorithmAmplitude = 0xFF;
0242            if (AlgorithmWidth < 0x08) {
0243              AlgorithmWidth++;
0244              TestAlgorithm_state = 0x05;
0245            } else {
0246              AlgorithmWidth = 0x02;
0247              if (AlgorithmChannelCount < 0x03){
0248                AlgorithmChannelCount++;
0249                switch (AlgorithmChannelCount){
0250                  case 0x00:
0251                    AlgorithmChannel = AlgTake1_Channel;
0252                    break;
0253                  case 0x01:
0254                    AlgorithmChannel = AlgTake2_Channel;
0255                    break;
0256                  case 0x02:
0257                    AlgorithmChannel = AlgTake3_Channel;
0258                    break;
0259                  case 0x03:
0260                    AlgorithmChannel = AlgTake4_Channel;
0261                    break;
0262                  default:
0263                    Serial.println("ERRR - Channel Update Error");
0264                    break;
0265                }
0266                TestAlgorithm_state = 0x02;
0267                break;
0268              } else {
0269                TestAlgorithm_state = 0x08;
0270                break;
0271              }
0272            }
0273          }
0274        }
0275        break;
0276      case 0x08: //
0277          if(errorCounter!=0){
0278            Serial.print("WARN - ");
0279            Serial.print(errorCounter);
0280            Serial.println(" errors in this run.");
0281          }
0282          Serial.println("INFO - Algorithm Finished!");
0283          TestAlgorithm_state = 0x00;
0284        break;
0285      default:
```

```
0286        Serial.println("ERRR - Algorithm State Machine Error!");
0287        TestAlgorithm_state = 0x00;
0288        break;
0289    }
0290 }
0291
0292 unsigned int DACregister(byte channel){
0293   byte low_channel = channel & 0x0F;
0294   byte hig_channel = channel & 0xF0;
0295   byte conv_register = 0x00;
0296   switch(low_channel){
0297     case 0x00:
0298       conv_register = 0x03;
0299       break;
0300     case 0x01:
0301       conv_register = 0x08;
0302       break;
0303     case 0x02:
0304       conv_register = 0x02;
0305       break;
0306     case 0x03:
0307       conv_register = 0x09;
0308       break;
0309     case 0x04:
0310       conv_register = 0x07;
0311       break;
0312     case 0x05:
0313       conv_register = 0x0C;
0314       break;
0315     case 0x06:
0316       conv_register = 0x06;
0317       break;
0318     case 0x07:
0319       conv_register = 0x0D;
0320       break;
0321     case 0x08:
0322       conv_register = 0x05;
0323       break;
0324     case 0x09:
0325       conv_register = 0x0E;
0326       break;
0327     case 0x0A:
0328       conv_register = 0x04;
0329       break;
0330     case 0x0B:
0331       conv_register = 0x0F;
0332       break;
0333     case 0x0C:
0334       conv_register = 0x01;
0335       break;
0336     case 0x0D:
0337       conv_register = 0x0A;
0338       break;
0339     case 0x0E:
0340       conv_register = 0x00;
0341       break;
0342     case 0x0F:
0343       conv_register = 0x0B;
0344       break;
0345     default:
0346       conv_register = 0x00;
0347       break;
0348   }
0349   return conv_register | hig_channel;
0350 }
0351
0352 void I2CMaskAllOff(){
0353   InitialAddr=0x0800;
0354   InitiateI2C();
0355   Serial.println("Mask All Off");
0356   for (int i=0; i < 16; i++){
0357     Wire.write(0x00);
```

```
0358   }
0359   Wire.endTransmission();
0360   InitialAddr=0x0810;
0361   InitiateI2C();
0362   for (int i=0; i < 16; i++){
0363     Wire.write(0x00);
0364   }
0365   Wire.endTransmission();
0366 }
0367
0368 void SimpleI2Cwrite(unsigned int RegAddr, byte RegData, boolean print){
0369   InitialAddr=RegAddr;
0370   InitiateI2C();
0371   Wire.write(RegData);
0372   Wire.endTransmission();
0373   if (print) {
0374     Serial.print("TX 0x");
0375     if (RegData < 0x10) Serial.print("0");
0376     Serial.print(RegData, HEX);
0377     Serial.print(" @ 0x");
0378     if (InitialAddr < 0x1000) Serial.print("0");
0379     if (InitialAddr < 0x100) Serial.print("0");
0380     if (InitialAddr < 0x10) Serial.print("0");
0381     Serial.print(InitialAddr, HEX);
0382   }
0383 }
0384
0385 byte SimpleI2Cread(unsigned int RegAddr, boolean print){
0386   InitialAddr=RegAddr;
0387   InitiateI2C();
0388   Wire.endTransmission();
0389   Wire.flush();
0390   unsigned int RegsToRead = 1;
0391   Wire.requestFrom(SlaveAddress, RegsToRead);
0392   byte received=Wire.read();
0393   if (print) {
0394     Serial.print(" :: RX 0x");
0395     if (received < 0x10) Serial.print("0");
0396     Serial.print(received, HEX);
0397     Serial.print(" @ 0x");
0398     if (InitialAddr < 0x1000) Serial.print("0");
0399     if (InitialAddr < 0x100) Serial.print("0");
0400     if (InitialAddr < 0x10) Serial.print("0");
0401     Serial.println(InitialAddr, HEX);
0402   }
0403   return received;
0404 }
0405
0406 void UserStateMachine() {
0407 //User Interface State Machine:
0408 byte inChar=(byte)Serial.read();  // Read next byte of UART FIFO
0409
0410 // If the user has pressed <ENTER>
0411 if(inChar==0x0D) {
0412   switch (state) {
0413     case 0x01:       // <ENTER> on write mode step 1 (after enter Initial Reg Add)
0414       InitiateI2C();
0415       halfByte=true;
0416       TxSeq = 0x00;
0417       state=0x02;
0418       Serial.println();
0419       Serial.println("Enter data to TX (XXh):");
0420       PrintRegIndex(TxSeq);
0421       break;
0422     case 0x03:       // <ENTER> on read mode step 1 (after enter Initial Reg Add)
0423       state=0x04;
0424       Serial.println();
0425       Serial.print("Number of addresses to read (XXh): ");
0426       break;
0427     case 0x05:
0428       state=0x06;
0429       Serial.println();
```

```
0430        Serial.print("Set test run identifier (XXXXh): ");
0431        break;
0432      case 0x07:
0433        TestNumber = InitialAddr;
0434        InitialAddr = 0x0000;
0435        NumberAddr = 0x0000;
0436        Serial.println();
0437        Serial.print("Set 1st channel to acquire (XXh): ");
0438        state = 0x08;   // Jump to Initial Chanel Set State
0439        break;
0440      case 0x02:       // <ENTER> on write mode step 2 (after enter all data to TX)
0441        Serial.println();
0442        Serial.println();
0443        TxCicle();
0444        state = 0x00;   // Jump to idle state
0445        break;
0446      case 0x04:       // <ENTER>  on read mode step 2 (after enter number of regs to RX)
0447        Serial.println();
0448        ReadI2C(true);
0449        state = 0x00;   // Jump to idle state
0450        break;
0451      case 0x06:
0452        Serial.println();
0453        TestChannel = InitialAddr;
0454        //TestIdentifier = "TEST";
0455        goState = true;
0456        state=0x00;
0457        break;
0458      case 0x08:
0459        Serial.println();
0460        AlgTake1_Channel = NumberAddr;
0461        NumberAddr = 0x0000;
0462        Serial.print("Set 2nd channel to acquire (XXh): ");
0463        state = 0x20;   // Jump to idle state
0464        break;
0465      case 0x20:
0466        Serial.println();
0467        AlgTake2_Channel = NumberAddr;
0468        NumberAddr = 0x0000;
0469        Serial.print("Set 3nd channel to acquire (XXh): ");
0470        state = 0x21;   // Jump to idle state
0471        break;
0472      case 0x21:
0473        Serial.println();
0474        AlgTake3_Channel = NumberAddr;
0475        NumberAddr = 0x0000;
0476        Serial.print("Set 4nd channel to acquire (XXh): ");
0477        state = 0x22;   // Jump to idle state
0478        break;
0479      case 0x22:
0480        Serial.println();
0481        AlgTake4_Channel = NumberAddr;
0482        NumberAddr = 0x0000;
0483        Serial.print("Amplitude scan step size (XXh): ");
0484        state = 0x23;   // Jump to idle state
0485        break;
0486      case 0x23:
0487        Serial.println();
0488        AlgorithmAmplitudeStep = NumberAddr;
0489        NumberAddr = 0x0000;
0490        Serial.print("Acquire time in seconds (XXh): ");
0491        state = 0x24;   // Jump to idle state
0492        break;
0493      case 0x24:
0494        Serial.println();
0495        DURATION_OF_AQUISITION = NumberAddr*1000;
0496        NumberAddr = 0x0000;
0497        state = 0x25;   // Jump to idle state
0498        Serial.println("Press <ENTER> to start");
0499        break;
0500      case 0x25:
0501        goTestAlgorithm = true;
```

```
0502        Serial.println("INFO - Test Algorithm Started");
0503        Serial.print("INFO - Channels (decimal): ");
0504        Serial.print(AlgTake1_Channel);
0505        Serial.print(", ");
0506        Serial.print(AlgTake2_Channel);
0507        Serial.print(", ");
0508        Serial.print(AlgTake3_Channel);
0509        Serial.print(", ");
0510        Serial.println(AlgTake4_Channel);
0511        Serial.print("INFO - Acquire time in ms (decimal): ");
0512        Serial.println(DURATION_OF_AQUISITION);
0513        state = 0x00;   // Jump to idle state
0514        break;
0515      case 0xF0:
0516        SlaveAddress = InitialAddr;
0517        Serial.println();
0518        Serial.print("INFO - Slave Address set to: 0x");
0519        Serial.println(SlaveAddress, HEX);
0520        state = 0x00;   // Jump to idle state
0521        break;
0522      default:
0523        CommandList();
0524    }
0525    return;
0526  }
0527
0528  if(TestAlgorithm_state > 0x00||inChar==0x1B){
0529    AbortTestAlgorithm = true;
0530  }
0531
0532  // If user press any key on write mode step 2 (enter Data to TX):
0533  if(state==0x2) {
0534    if(inChar==0x1B) { // ABORT if press ESC
0535      state=0x0;
0536      AbortInfo();
0537    }
0538    byte converted = convert(inChar);
0539    if(illegalInput) return;
0540    Serial.print(converted, HEX);
0541    if(halfByte)
0542    {
0543      next_write = convert(inChar);
0544      halfByte=false;
0545    } else {
0546      next_write = next_write << 4;
0547      next_write = next_write|converted;
0548      Wire.write(next_write);
0549      Serial.print(" ");
0550      halfByte=true;
0551      TxSeq++;
0552    }
0553    if ((TxSeq&0x000F)==0x0000&&halfByte==true)
0554    {
0555      Serial.println();
0556      TxCicle();
0557      TxSeq=0x00;
0558      InitialAddr=InitialAddr+0x010;
0559      InitiateI2C();
0560      PrintRegIndex(TxSeq);
0561    }
0562    return;
0563  }
0564
0565  // If user press any key on read mode step 2 (number of addr to read):
0566
if(state==0x04||state==0x06||state==0x08||state==0x20||state==0x21||state==0x22||state==0x23||
state==0x24) {
0567    if(inChar==0x1B) { // ABORT if press ESC
0568      state=0x00;
0569      AbortInfo();
0570    }
0571    byte converted = convert(inChar);
```

```
0572   if(illegalInput) return;
0573   Serial.print(converted, HEX);
0574   NumberAddr = NumberAddr << 4;
0575   NumberAddr = NumberAddr|converted;
0576   return;
0577 }
0578
0579 // If user press any key on another number entering mode:
0580 if(state==0x01||state==0x03||state==0x05||state==0x07||state==0xF0) {
0581   if(inChar==0x1B) { // ABORT if press ESC
0582     state=0x00;
0583     AbortInfo();
0584   }
0585   byte converted = convert(inChar);
0586   if(illegalInput) return;
0587   Serial.print(converted, HEX);
0588   InitialAddr = InitialAddr << 4;
0589   InitialAddr = InitialAddr|converted;
0590   return;
0591 }
0592
0593 // If receives a 'T': Start the test procedure!
0594 if(inChar==0x54||inChar==0x74)
0595 {
0596   if(state!=0x0) AbortInfo();
0597   state = 0x05;
0598   InitialAddr = 0x0000;
0599   NumberAddr = 0x0000;
0600   Serial.print("Test Procedure. Set channel (XXXXh): ");
0601   return;
0602 }
0603
0604 // If receives a 'Y': Start test algorithm!
0605 if(inChar==0x59||inChar==0x79)
0606 {
0607   if(state!=0x0) AbortInfo();
0608   state=0x07;
0609   InitialAddr = 0x0000;
0610   NumberAddr = 0x0000;
0611   Serial.println();
0612   Serial.print("Set test run number (XXXXh): ");
0613
0614   return;
0615 }
0616
0617 // If receives a 'W': Go to write mode (0x1)
0618 if(inChar==0x57||inChar==0x77)
0619 {
0620   if(state!=0x0) AbortInfo();
0621   state = 0x01;
0622   InitialAddr = 0x0000;
0623   Serial.print("Write mode. Set initial address (XXXXh): ");
0624   return;
0625 }
0626
0627 // If receives a 'R': Go to read mode (0x3)
0628 if(inChar==0x52||inChar==0x72)
0629 {
0630   if(state!=0x00) AbortInfo();
0631   state = 0x03;
0632   InitialAddr = 0x0000;
0633   NumberAddr = 0x0000;
0634   Serial.print("Read mode. Set initial address (XXXXh): ");
0635   return;
0636 }
0637
0638 // If receives a 'A': Go to address change mode (0xF0)
0639 if(inChar==0x41||inChar==0x61)
0640 {
0641   if(state!=0x00) AbortInfo();
0642   state = 0xF0;
0643   InitialAddr = 0x0000;
```

```
0644   Serial.print("Enter slave address (XXh): ");
0645   return;
0646 }
0647
0648 // If receives a 'S': Do a I2C address scan
0649 if(inChar==0x53||inChar==0x73)
0650 {
0651   I2Cscan();
0652 }
0653 }
0654
0655
0656 void TestProcStateMachine() {
0657 // Test Procedure State Machine:
0658   switch (test_state)
0659   {
0660     case 0x00: // IDLE
0661       if (goState == true) {
0662       goState = false;
0663       test_state = 0x01;
0664       }
0665       break;
0666     case 0x01:
0667       start_timestamp = millis();
0668       startAquisition();
0669       Serial.print("INFO - Routine ");
0670       Serial.print(TestIdentifier);
0671       Serial.println(" Started...");
0672       Serial.print("TASK - DATA-ACQ_start>000");
0673       acq_counter = 1;
0674       test_state = 0x02;
0675       break;
0676     case 0x02:
0677       if (millis() >= start_timestamp + DURATION_OF_AQUISITION) {
0678         stopAquisition();
0679         Serial.write(0x08);
0680         Serial.write(0x08);
0681         Serial.write(0x08);
0682         if (acq_counter<100) Serial.print("0");
0683         if (acq_counter<10) Serial.print("0");
0684         Serial.print(acq_counter);
0685         Serial.print(">stop");
0686         test_state = 0x03;
0687         volatile_timestamp = millis();
0688       }
0689       else if (millis() >= (start_timestamp + (acq_counter*1000))) {
0690         Serial.write(0x08);
0691         Serial.write(0x08);
0692         Serial.write(0x08);
0693         if (acq_counter<100) Serial.print("0");
0694         if (acq_counter<10) Serial.print("0");
0695         Serial.print(acq_counter);
0696         acq_counter++;
0697       }
0698       break;
0699     case 0x03:
0700       if (millis() >= volatile_timestamp + responseDelay*2) {
0701         Serial.print(" :: SCREEN_saving");
0702         saveScreen();
0703         Serial.write(0x08);
0704         Serial.write(0x08);
0705         Serial.write(0x08);
0706         Serial.write(0x08);
0707         Serial.write(0x08);
0708         Serial.write(0x08);
0709         Serial.print("saved ");
0710         test_state = 0x04;
0711         volatile_timestamp = millis();
0712       }
0713       break;
0714     case 0x04:
0715       if (millis() >= volatile_timestamp + responseDelay*6) {
```

```
0716          Serial.print(":: MEASUR_saving");
0717          saveMeasurements();
0718          Serial.write(0x08);
0719          Serial.write(0x08);
0720          Serial.write(0x08);
0721          Serial.write(0x08);
0722          Serial.write(0x08);
0723          Serial.write(0x08);
0724          Serial.println("saved !");
0725          test_state = 0x05;
0726          volatile_timestamp = millis();
0727        }
0728        break;
0729      case 0x05:
0730        if (millis() >= volatile_timestamp + responseDelay*2) {
0731          Serial.print("INFO - Routine ");
0732          Serial.print(TestIdentifier);
0733          Serial.println(" Finished !!");
0734          test_state = 0x00;
0735          endState=true;
0736        }
0737        break;
0738      default:
0739        Serial.println("ERRR - Test State Machine Error!");
0740        test_state = 0x00;
0741        endState=true;
0742        break;
0743    }
0744 }
0745
0746 void debugAquisition() {
0747   Keyboard.write(KEY_F6);
0748   delay(responseDelay);
0749   Keyboard.write(KEY_F5);
0750   delay(responseDelay);
0751   Keyboard.write(KEY_F5);
0752   delay(responseDelay);
0753   Keyboard.write(KEY_F6);
0754 }
0755
0756 void startAquisition() {
0757   Keyboard.press(KEY_LEFT_ALT);
0758   Keyboard.write('c');
0759   delay(responseDelay);
0760   Keyboard.releaseAll();
0761   delay(responseDelay);
0762   Keyboard.write(KEY_F5);
0763 }
0764
0765 void stopAquisition() {
0766   Keyboard.write(KEY_F6);
0767 }
0768
0769 void saveScreen() {
0770   Keyboard.write(KEY_LEFT_ALT);
0771   delay(responseDelay);
0772   Keyboard.write('f');
0773   delay(responseDelay);
0774   Keyboard.write('s');
0775   delay(responseDelay);
0776   Keyboard.write('c');
0777   delay(responseDelay*4);
0778   Keyboard.print(TestIdentifier);
0779   delay(responseDelay);
0780   Keyboard.write(KEY_TAB);
0781   delay(responseDelay);
0782   Keyboard.write('p');
0783   delay(responseDelay);
0784   Keyboard.write(KEY_RETURN);
0785 }
0786
0787 void saveMeasurements() {
```

```
0788   Keyboard.write(KEY_LEFT_ALT);
0789   delay(responseDelay);
0790   Keyboard.write('f');
0791   delay(responseDelay);
0792   Keyboard.write('s');
0793   delay(responseDelay);
0794   Keyboard.write('m');
0795   delay(responseDelay*4);
0796   Keyboard.print(TestIdentifier);
0797   delay(responseDelay);
0798   Keyboard.write(KEY_RETURN);
0799 }
0800
0801 void loadSetup(byte SetupType , byte SetupChannel){
0802   String file_name = "";
0803   if (SetupType==0x00){
0804     file_name = "XTALK";
0805   } else {
0806     file_name = "INPUT";
0807   }
0808   file_name = file_name + (String)SetupChannel;
0809
0810   Keyboard.write(KEY_LEFT_ALT);
0811   delay(responseDelay);
0812   Keyboard.write('f');
0813   delay(responseDelay);
0814   Keyboard.write('l');
0815   delay(responseDelay);
0816   Keyboard.write('s');
0817   delay(responseDelay);
0818   Keyboard.print(file_name);
0819   delay(responseDelay);
0820   Keyboard.write(KEY_RETURN);
0821   delay(responseDelay*16);
0822 }
0823
0824 byte ComputeTxCRC(byte initial)
0825 {
0826   byte remainder = 0;
0827   for (int n_byte = initial; n_byte < BufferLength; ++n_byte)
0828   {
0829     remainder ^= Wire.ReadTxBuffer(n_byte);
0830     for (uint8_t bit = 8; bit > 0; --bit)
0831     {
0832       if (remainder & 0x80)
0833       {
0834           remainder = (remainder << 1) ^ 0xD8;
0835       }
0836       else
0837       {
0838           remainder = (remainder << 1);
0839       }
0840     }
0841   }
0842   return (remainder);
0843 }
0844
0845 byte ComputeRxCRC(byte initial)
0846 {
0847   byte remainder = 0;
0848   for (int n_byte = initial; n_byte < BufferLength; ++n_byte)
0849   {
0850     remainder ^= Wire.ReadRxBuffer(n_byte);
0851     for (uint8_t bit = 8; bit > 0; --bit)
0852     {
0853       if (remainder & 0x80)
0854       {
0855           remainder = (remainder << 1) ^ 0xD8;
0856       }
0857       else
0858       {
0859           remainder = (remainder << 1);
```

```
0860       }
0861     }
0862   }
0863   return (remainder);
0864 }
0865 void InitiateI2C()
0866 {
0867   Wire.beginTransmission(SlaveAddress);
0868   Wire.write(lowByte(InitialAddr));
0869   Wire.write(highByte(InitialAddr));
0870 }
0871
0872 void PrintRegIndex(unsigned int partial)
0873 {
0874   unsigned int LineRegAdd = InitialAddr+partial;
0875   Serial.println();
0876   if (LineRegAdd < 0x1000) Serial.print("0");
0877   if (LineRegAdd < 0x100) Serial.print("0");
0878   if (LineRegAdd < 0x10) Serial.print("0");
0879   Serial.print(LineRegAdd, HEX);
0880   Serial.print(": ");
0881 }
0882
0883 void TxCicle()
0884 {
0885   BufferLength = Wire.getTxBufferLength();
0886   CRCtx = ComputeTxCRC(0x02);
0887   Serial.print("TASK - I2C TX: ");
0888   Serial.print(BufferLength);
0889   Serial.print(" bytes - 1st Addr: 0x");
0890   if (InitialAddr < 0x1000) Serial.print("0");
0891   if (InitialAddr < 0x100) Serial.print("0");
0892   if (InitialAddr < 0x10) Serial.print("0");
0893   Serial.print(InitialAddr, HEX);
0894   Serial.print(" CRC: 0x");
0895   if (CRCtx < 0x10) Serial.print("0");
0896   Serial.println(CRCtx, HEX);
0897   Wire.endTransmission();
0898 }
0899
0900 void I2Cscan()
0901 {
0902   byte I2CAddressScan = 0;
0903   Serial.println("INFO - Slave scan start...");
0904   while(I2CAddressScan < 127)
0905   {
0906     I2CAddressScan++;    // 0x00 are a "General Call" address, so we skip it
0907     Wire.beginTransmission(I2CAddressScan);
0908     if (Wire.endTransmission() == 0)
0909     {
0910       SlaveAddress = I2CAddressScan;
0911       Serial.print("INFO - Found Slave: 0x");
0912       Serial.println(I2CAddressScan, HEX);
0913     }
0914   }
0915   Serial.println("INFO - Slave scan end !");
0916
0917   if (SlaveAddress == 0)
0918   {
0919     Serial.println("ERRR - No device found! Please check your hardware");
0920     Serial.println("INFO - Press 'S' to try again");
0921   }
0922   else
0923   {
0924     Serial.print("INFO - Slave Address set to: 0x");
0925     Serial.println(SlaveAddress, HEX);
0926   }
0927 }
0928
0929
0930 void AbortInfo()
0931 {
```

```
0932   Serial.println("");
0933   Serial.println("INFO - Current operation aborted");
0934 }
0935
0936 byte convert(byte input)
0937 {
0938   illegalInput=false;
0939   byte output;
0940   switch (input)
0941   {
0942   case 0x30:
0943     output=0x0;
0944     break;
0945   case 0x31:
0946     output=0x1;
0947     break;
0948   case 0x32:
0949     output=0x2;
0950     break;
0951   case 0x33:
0952     output=0x3;
0953     break;
0954   case 0x34:
0955     output=0x4;
0956     break;
0957   case 0x35:
0958     output=0x5;
0959     break;
0960   case 0x36:
0961     output=0x6;
0962     break;
0963   case 0x37:
0964     output=0x7;
0965     break;
0966   case 0x38:
0967     output=0x8;
0968     break;
0969   case 0x39:
0970     output=0x9;
0971     break;
0972   case 0x41:
0973     output=0xA;
0974     break;
0975   case 0x42:
0976     output=0xB;
0977     break;
0978   case 0x43:
0979     output=0xC;
0980     break;
0981   case 0x44:
0982     output=0xD;
0983     break;
0984   case 0x45:
0985     output=0xE;
0986     break;
0987   case 0x46:
0988     output=0xF;
0989     break;
0990   case 0x61:
0991     output=0xA;
0992     break;
0993   case 0x62:
0994     output=0xB;
0995     break;
0996   case 0x63:
0997     output=0xC;
0998     break;
0999   case 0x64:
1000     output=0xD;
1001     break;
1002   case 0x65:
1003     output=0xE;
```

```
1004      break;
1005    case 0x66:
1006      output=0xF;
1007      break;
1008    default:
1009      illegalInput=true;
1010      break;
1011    }
1012    return output;
1013  }
1014
1015  void ReadI2C(boolean Print)
1016  {
1017    InitiateI2C();
1018    Wire.endTransmission();
1019    Wire.flush();
1020    Wire.requestFrom(SlaveAddress, NumberAddr+1);
1021    BufferLength = Wire.getRxBufferLength();
1022    CRCrx = ComputeRxCRC(0x00);
1023    if(Print) {
1024      byte RxSeq = 0x00;
1025      while (Wire.available()) {
1026        byte received=Wire.read();
1027        if ((RxSeq&0x0F)==0x00) PrintRegIndex(RxSeq);
1028        if (received < 0x10) Serial.print("0");
1029        Serial.print(received, HEX);
1030        Serial.print(" ");
1031        RxSeq++;
1032      }
1033      Serial.println();
1034      Serial.print("TASK - I2C RX: ");
1035      Serial.print(BufferLength);
1036      Serial.print(" reg(s) - 1st: 0x");
1037      if (InitialAddr < 0x1000) Serial.print("0");
1038      if (InitialAddr < 0x100) Serial.print("0");
1039      if (InitialAddr < 0x10) Serial.print("0");
1040      Serial.print(InitialAddr, HEX);
1041      Serial.print(" CRC: 0x");
1042      if (CRCrx < 0x10) Serial.print("0");
1043      Serial.println(CRCrx, HEX);
1044      Serial.println();
1045    }
1046  }
1047
1048  void CommandList()
1049  {
1050    Serial.println("INFO - Command list:");
1051    Serial.println("       'S' -> Do a slave scan");
1052    Serial.println("       'A' -> Change slave address (7-bit)");
1053    Serial.println("       'W' -> Write data on Slave");
1054    Serial.println("       'R' -> Read up to 256 bytes from Slave");
1055    Serial.println("       'T' -> Do a test routine");
1056    Serial.println("       'Y' -> Start test Algorithm");
1057    Serial.println("       ESC -> Abort current operation");
1058  }
```

Apêndice I

Script Python para Formatação dos Dados

```
001 # coding=UTF-8
002
003 import os
004 from operator import itemgetter
005
006 source_dir  = 'RAW'
007 output_file = 'SOURCE/DATAMASTER test.csv'
008 space_delimited = False
009 # NumberOfCharactersOnIndex
010 n = 78
011
012 iDir   = 0
013 iFiles = 0
014 iFound = 0
015 iFileRead = 0
016
017
018 print " ---------- PROCESSING STARTED ------------ "
019
020 ScopeFiles=[]
021 for (dirpath, dirnames, filenames) in os.walk(source_dir):
022     for fname in filenames:
023         if fname[-4:] == ".csv":
024             ScopeFiles.append([])
025             Channel = int("0x%s" % (fname[8:10]), 0)
026             Width   = int("0x%s" % (fname[11:13]), 0)
027             DAC     = int("0x%s" % (fname[14:16]), 0)
028             Type    = fname[5]
029             ID      = Channel*10000000+Width*1000+(255-DAC)*1
030             if Type == "i": ID += 1000000
031             ScopeFiles[iFound].append("%s/%s" % (dirpath, fname))
032             ScopeFiles[iFound].append(Type)
033             ScopeFiles[iFound].append(Channel)
034             ScopeFiles[iFound].append(Width)
035             ScopeFiles[iFound].append(DAC)
036             ScopeFiles[iFound].append(ID)
037             iFound += 1
038         iFiles += 1
039     iDir += 1
040
041
042 print "number of directories found           = ",iDir
043 print "number of files found                 = ",iFiles
044 print "number of csv measurement files found  = ",iFound
045
046 ScopeFiles.sort(key=itemgetter(5))
047
048
049 # Open the output file in append mode:
050 with open(output_file, 'w') as outputfile:
051
052     # Write the header (1st line of output file):
053     outputfile.write("Typ,Cha,Wid,Amp,"+\
054                      "Measurement1,Mean1,Unit,Min1,Unit,Max1,Unit,
    Range1,Unit,Sigma1,Unit,NumberOfMeas1,Unit,"+\
055                      "Measurement2,Mean2,Unit,Min2,Unit,Max2,Unit,
    Range2,Unit,Sigma2,Unit,NumberOfMeas2,Unit,"+\
056                      "Measurement3,Mean3,Unit,Min3,Unit,Max3,Unit,
    Range3,Unit,Sigma3,Unit,NumberOfMeas3,Unit,"+\
057                      "Measurement4,Mean4,Unit,Min4,Unit,Max4,Unit,
    Range4,Unit,Sigma4,Unit,NumberOfMeas4,Unit,"+\
```

```
058                     "Measurement5,Mean5,Unit,Min5,Unit,Max5,Unit,
      Range5,Unit,Sigma5,Unit,NumberOfMeas5,Unit\n")
059
060     # Read each line of file_index file ...
061     for fileline in ScopeFiles:
062
063         iFileRead += 1
064         arquivoatual = fileline[0].rstrip()
065         lineindex = 1
066
067         # ... and recursively open each scope csv file listed on
file_index:
068         with open(arquivoatual) as csvdata:
069             for dataline in csvdata:
070                 dados = dataline.rstrip()
071                 if lineindex == 5:
072                     name1 = dados[16:31]
073                     name2 = dados[32:47]
074                     name3 = dados[48:63]
075                     name4 = dados[64:79]
076                     name5 = dados[80:95]
077                 if lineindex == 7:
078                     mean1 = dados[16:31]
079                     mean2 = dados[32:47]
080                     mean3 = dados[48:63]
081                     mean4 = dados[64:79]
082                     mean5 = dados[80:95]
083                 if lineindex == 8:
084                     min1 = dados[16:31]
085                     min2 = dados[32:47]
086                     min3 = dados[48:63]
087                     min4 = dados[64:79]
088                     min5 = dados[80:95]
089                 if lineindex == 9:
090                     max1 = dados[16:31]
091                     max2 = dados[32:47]
092                     max3 = dados[48:63]
093                     max4 = dados[64:79]
094                     max5 = dados[80:95]
095                 if lineindex == 10:
096                     range1 = dados[16:31]
097                     range2 = dados[32:47]
098                     range3 = dados[48:63]
099                     range4 = dados[64:79]
100                     range5 = dados[80:95]
101                 if lineindex == 11:
102                     sigma1 = dados[16:31]
103                     sigma2 = dados[32:47]
104                     sigma3 = dados[48:63]
105                     sigma4 = dados[64:79]
106                     sigma5 = dados[80:95]
107                 if lineindex == 12:
108                     n1 = dados[16:31]
109                     n2 = dados[32:47]
110                     n3 = dados[48:63]
111                     n4 = dados[64:79]
112                     n5 = dados[80:95]
113                 lineindex += 1
114             outputdataline = "%s,0x%02X,0x%02X,0x%02X,\'%s\',
      %s,%s,%s,%s,%s,%s,pts,\'%s\',%s,%s,%s,%s,%s,%s,pts,\'%s\',
      %s,%s,%s,%s,%s,%s,pts,\'%s\',%s,%s,%s,%s,%s,%s,pts,\'%s\',
```

```
    %s,%s,%s,%s,%s,%s,pts\n" % \
115                                 (fileline[1],fileline[2],fileline[3],
    fileline[4],\
116                                  name1, mean1, min1, max1, range1,
     sigma1, n1,\
117                                  name2, mean2, min2, max2, range2,
     sigma2, n2,\
118                                  name3, mean3, min3, max3, range3,
     sigma3, n3,\
119                                  name4, mean4, min4, max4, range4,
     sigma4, n4,\
120                                  name5, mean5, min5, max5, range5,
     sigma5, n5)
121             outputdataline = outputdataline.replace(" ", "")
122             outputdataline = outputdataline.replace("pVs,", "E-12,Vs,")
123             outputdataline = outputdataline.replace("fVs,", "E-15,Vs,")
124             outputdataline = outputdataline.replace("0.0Vs,",
     "0.0,Vs,")
125             outputdataline = outputdataline.replace("k,", "E+03,")
126             outputdataline = outputdataline.replace("ns,", "E-09,s,")
127             outputdataline = outputdataline.replace("ps,", "E-12,s,")
128             outputdataline = outputdataline.replace("0.0s,", "0.0,s,")
129             outputdataline = outputdataline.replace("mV,", "E-03,V,")
130             outputdataline = outputdataline.replace("µV,", "E-06,V,")
131             outputdataline = outputdataline.replace("nV,", "E-09,V,")
132             outputdataline = outputdataline.replace("0.0V,", "0.0,V,")
133             outputdataline = outputdataline.replace("---------------,
    ---------------,---------------,---------------,---------------,
    ---------------,", "0,---,0,---,0,---,0,---,0,---,0,")
134             outputdataline = outputdataline.replace("?", "")
135             if space_delimited:
136                 outputdataline = outputdataline.replace(",", " ")
137             outputfile.write(outputdataline)
138
139
140             # Print partial processing progress:
141             if iFileRead % 500 == 0 :
142                 print "processed", iFileRead," input files "
143
144
145 print " ------------ PROCESSING END --------------- "
146 print "number of measurement files read  = ",iFileRead
```

Apêndice J

Script Python para Análise dos Dados

```
0000 import ROOT
0001 import os
0002 import sys
0003
0004 from datetime import datetime
0005 from operator import itemgetter
0006 from math import *
0007 from array import array
0008 from ROOT import TFile, TTree, TH1F, TCanvas, TGraphErrors, TF1, TMultiGraph,\
0009                 TLegend, gPad, TColor, TH1, TLatex, TPaveText, gROOT
0010
0011 versionid   = "RUN01"
0012 nametree    = "data_"+versionid+".root"
0013 nbin        = 30
0014 mintiming   = 559
0015 maxtiming   = 564
0016 NumberOfAmplitudes = 4
0017
0018 PlotChLimit = 1000
0019 HistLimit   = 1000
0020
0021 # Enable SingleCanvas to reuse the same canvas for a specific kind of chart
0022 GfxSingleCanvas = True
0023 HisSingleCanvas = False
0024
0025 ExportPdf = True
0026 ExportEps = True
0027
0028 impedance   = 50 #Ohms
0029
0030 StartTimeStamp = datetime.now().strftime('%Y-%m-%d_%Hh%Mm%Ss')
0031
0032 sfile_datamaster = "SOURCE/DATAMASTER.csv"
0033 Log_file         = "Log_%s.txt" % (StartTimeStamp)
0034 Script_file      = "Script_%s.py" % (StartTimeStamp)
0035 ChWdFit_file     = "ChWdFit_%s.txt" % (StartTimeStamp)
0036 ChAmpFit_file    = "ChAmpFit_%s.txt" % (StartTimeStamp)
0037
0038 MeasNames = ['Area','Charge','Timing','Width','Peak']
0039
0040 EpsDirectory = "OUTPUT/%s/eps" % (StartTimeStamp)
0041 PdfDir = "OUTPUT/%s/pdf" % (StartTimeStamp)
0042 LogDirectory = "OUTPUT/%s/log" % (StartTimeStamp)
0043
0044 ### Configure ROOT to display statistics TBox:
0045 ROOT.gStyle.SetOptFit(111)
0046
0047 ### Configure ROOT to run in batch mode (disable screen output):
0048 gROOT.SetBatch(1)
0049
0050 #### Injector Mesurements Data
0051
0052
0053 # Initialize Fitting Data Structure:
0054 _Data=[]
0055 # 1st dimension = Data Column
0056 for column in range(0, 15):
0057     _Data.append([])
0058     # 2nd dimension = Data Row
0059     _Data[column]=array('d')
0060
0061 ### Old Data Structure Equivalent:
0062 #
0063 # _OutputChannel    = _Data[0]  = array('d')
0064 # _OutputWidth      = _Data[1]  = array('d')
0065 # _OutputAmplitude  = _Data[2]  = array('d')
0066 # _eOutputAmplitude = _Data[3]  = array('d')
0067 #
0068 # _PulseArea        = _Data[4]  = array('d')
0069 # _ePulseArea       = _Data[5]  = array('d')
```

```
0070 # _PulseCharge      = _Data[6]  = array('d')
0071 # _ePulseCharge     = _Data[7]  = array('d')
0072 # _PulseWidth       = _Data[8]  = array('d')
0073 # _ePulseWidth      = _Data[9]  = array('d')
0074 # _PulsePeak        = _Data[10] = array('d')
0075 # _ePulsePeak       = _Data[11] = array('d')
0076 # _PulseTiming      = _Data[12] = array('d')
0077 # _ePulseTiming     = _Data[13] = array('d')
0078
0079 nChMultiGfx = array( 'd' )
0080 wDACMultiGfx = array( 'd' )
0081 aDACMultiGfx = array( 'd' )
0082
0083 LogMessages = []
0084
0085 ##############################################################################
0086 ##                                                                          ##
0087 ##                        Definition of Script Functions                    ##
0088 ##                                                                          ##
0089 ##############################################################################
0090
0091 def LogIt(message):
0092     message = "%s - %s" % (datetime.now().strftime('%Y-%m-%d %H:%M:%S'), message)
0093     print message
0094     LogMessages.append(message)
0095
0096 # INPUT Data Structure:
0097 #
0098 #"Typ","Cha","Wid","Amp",
0099 #  0     1     2     3
0100 #"Measure1",Mean1,"Un",Min1,"Un",Max1,"Un",Range1,"Un",Sigma1,"Un",NumOfMeas1,"Un",
0101 #      4      5     6    7    8    9   10    11    12    13    14      15       16
0102 #"Measure2",Mean2,"Un",Min2,"Un",Max2,"Un",Range2,"Un",Sigma2,"Un",NumOfMeas2,"Un",
0103 #     17     18    19   20   21   22   23    24    25    26    27      28       29
0104 #"Measure3",Mean3,"Un",Min3,"Un",Max3,"Un",Range3,"Un",Sigma3,"Un",NumOfMeas3,"Un",
0105 #     30     31    32   33   34   35   36    37    38    39    40      41       42
0106 #"Measure4",Mean4,"Un",Min4,"Un",Max4,"Un",Range4,"Un",Sigma4,"Un",NumOfMeas4,"Un",
0107 #     43     44    45   46   47   48   49    50    51    52    53      54       55
0108 #"Measure5",Mean5,"Un",Min5,"Un",Max5,"Un",Range5,"Un",Sigma5,"Un",NumOfMeas5,"Un"
0109 #     56     57    58   59   60   61   62    63    64    65    66      67       68
0110
0111 def extract_DATAMASTER(c) :
0112
0113     list1 = c.split(",")
0114
0115 # OUTPUT Data Structure:
0116 #
0117 #"Typ",Cha,Wid,Amp,Mean1,Sigma1,Mean2,Sigma2,Mean3,Sigma3,Mean4,Sigma4,Mean5,Sigma5
0118 #  0    1   2   3    4      5     6      7     8      9     10    11    12     13
0119 #  Identificator  |  Area[Vs]  |  Timing[s] |  Width[s]  |  Peak[V]   | Baseline[V]
0120
0121     return list1[0],int(list1[1], 0),int(list1[2], 0),int(list1[3], 0),\
0122            float(list1[5]),float(list1[13]),float(list1[18]),float(list1[26]),\
0123            float(list1[31]),float(list1[39]),float(list1[44]),float(list1[52]),\
0124            float(list1[57]),float(list1[65])
0125
0126 ##################################################################
0127
0128 LogIt("------------       START      ------------")
0129
0130 ### loop over DATA
0131
0132 line_counter   = 0
0133 iacquisition         = 0
0134 isaved_data    = 0
0135
0136 input_file = open(sfile_datamaster)
0137
```

```
0138 for line in input_file :
0139
0140     line_counter += 1
0141
0142     # skiping header
0143     if line_counter > 1 :
0144         list_data = extract_DATAMASTER(line)
0145         data_type = list_data[0]
0146         iacquisition += 1
0147         # Do analysis only for input aquisition type data
0148         if data_type == "i" :
0149
0150             isaved_data += 1
0151
0152             InjectorChannel       = list_data[1]
0153             InjectorWidth         = list_data[2]
0154             InjectorAmplitude     = 255-list_data[3]
0155
0156             MeanMeasuredArea      = list_data[4]
0157             SigmaMeasuredArea     = list_data[5]
0158             MeanMeasuredTime      = list_data[6]
0159             SigmaMeasuredTime     = list_data[7]
0160             MeanMeasuredWidth     = list_data[8]
0161             SigmaMeasuredWidth    = list_data[9]
0162             MeanMeasuredPeak      = list_data[10]
0163             SigmaMeasuredPeak     = list_data[11]
0164             MeanMeasBaseline      = list_data[12]
0165             SigmaMeasBaseline     = list_data[13]
0166
0167             AdjustedMeanArea  = MeanMeasuredArea - (MeanMeasBaseline*0.000000040)
0168             AdjustedAreaError = SigmaMeasuredArea + (SigmaMeasBaseline*0.000000040)
0169
0170             CalculatedMeanCharge  = AdjustedMeanArea / impedance
0171             CalculatedChargeError = AdjustedAreaError / impedance
0172
0173             _Data[0].append(InjectorChannel)            ### Old _OutputChannel
0174
0175             _Data[1].append(InjectorWidth)              ### Old _OutputWidth
0176             _Data[14].append(0.0)                       ### NEW _eOutputWidth
0177
0178             _Data[2].append(InjectorAmplitude)          ### Old _OutputAmplitude
0179             _Data[3].append(0.0)                        ### Old _eOutputAmplitude
0180
0181             _Data[4].append(AdjustedMeanArea)           ### Old _PulseArea
0182             _Data[5].append(AdjustedAreaError)          ### Old _ePulseArea
0183
0184             _Data[6].append(CalculatedMeanCharge)       ### Old _PulseCharge
0185             _Data[7].append(CalculatedChargeError)      ### Old _ePulseCharge
0186
0187             _Data[8].append(MeanMeasuredWidth)          ### Old _PulseWidth
0188             _Data[9].append(SigmaMeasuredWidth)         ### Old _ePulseWidth
0189
0190             _Data[10].append(MeanMeasuredPeak)          ### Old _PulsePeak
0191             _Data[11].append(SigmaMeasuredPeak)         ### Old _ePulsePeak
0192
0193             _Data[12].append(MeanMeasuredTime)          ### Old _PulseTiming
0194             _Data[13].append(SigmaMeasuredTime)         ### Old _ePulseTiming
0195
0196             # Print partial processing progress:
0197             if iacquisition % 500 == 0 :
0198                 print "processed", iacquisition," acquisitions"
0199
0200 # Define number of Measurements, Channels, Widths and Amplitudes:
0201 _nM = len(MeasNames)
0202 _nC = len(set(_Data[0]))
0203 _nW = len(set(_Data[1]))
0204 _nA = len(set(_Data[2]))
0205
0206 _CountSet = []
0207 for i in range(0, 3):
```

```
0208     temp = 0
0209     count = 0
0210     _CountSet.append([])
0211     for value in set(_Data[i]):
0212         for row in _Data[i]:
0213             if value == row: temp += 1
0214         _CountSet[i].append([])
0215         _CountSet[i][count].append(value)
0216         _CountSet[i][count].append(temp)
0217         temp = 0
0218         count += 1
0219     _CountSet[i].sort(key=itemgetter(0))
0220
0221 LogIt("----------- DATAMASTER  READ ------------- ")
0222 LogIt("Nr of processed acquisitions = %11s"%(iacquisition))
0223 LogIt("Nr of saved acquisitions     = %11s"%(isaved_data))
0224 LogIt("------------ CONFIG OPTIONS -------------- ")
0225 LogIt("Amplitudes on Timing Dist    = %11s"%(NumberOfAmplitudes))
0226 LogIt("Multi Gfx Channel Limit      = %11s"%(PlotChLimit))
0227 LogIt("Multi Gfx Single Canvas      = %11s"%(GfxSingleCanvas))
0228 LogIt("Fit Histogram Single Canvas  = %11s"%(HisSingleCanvas))
0229 LogIt("Export to PDF file           = %11s"%(ExportPdf))
0230 LogIt("Export to EPS file           = %11s"%(ExportEps))
0231 LogIt("----------- DATASTRUCTURE INFO ------------ ")
0232 LogIt("Number of Measurements       = %11s"%(_nM))
0233 LogIt("Number of Channels           = %11s"%(_nC))
0234 LogIt("Number of Widths             = %11s"%(_nW))
0235 LogIt("Number of Amplitudes         = %11s"%(_nA))
0236 LogIt("------------ CHANNELS  LIST -------------- ")
0237 for line in _CountSet[0]:
0238     LogIt("Channel   = %9s found %8s times"%(line[0], line[1]))
0239 LogIt("------------ WIDTHS  LIST ---------------- ")
0240 for line in _CountSet[1]:
0241     LogIt("Width     = %9s found %8s times"%(line[0], line[1]))
0242 LogIt("----------- AMPLITUDES  LIST ------------- ")
0243 for line in _CountSet[2]:
0244     LogIt("Amplitude = %9s found %8s times"%(line[0], line[1]))
0245
0246 #### Verify directory structure
0247 ## EPS Files:
0248 for m in range(0, _nM):
0249     DirPath1 = "%s/%s/WidScan" % (EpsDirectory, MeasNames[m])
0250     DirPath2 = "%s/%s/AmpScan" % (EpsDirectory, MeasNames[m])
0251     if not os.path.exists(DirPath1):
0252         os.makedirs(DirPath1)
0253         LogIt("INFO - Directory %s created" % (DirPath1))
0254     if not os.path.exists(DirPath2):
0255         os.makedirs(DirPath2)
0256         LogIt("INFO - Directory %s created" % (DirPath2))
0257
0258 ## PDF Files:
0259 DirPath = "%s" % (PdfDir)
0260 if not os.path.exists(DirPath):
0261     os.makedirs(DirPath)
0262     LogIt("INFO - Directory %s created" % (DirPath))
0263
0264 ## LOG File:
0265 DirPath = "%s" % (LogDirectory)
0266 if not os.path.exists(DirPath):
0267     os.makedirs(DirPath)
0268     LogIt("INFO - Directory %s created" % (DirPath))
0269
0270 # Initialize Amp Scan Fitting Data Structure:
0271 _ChWdFit=[]
0272 # 1st dimension = Measurement
0273 for m in range(0, _nM):
0274     _ChWdFit.append([])
0275     # 2nd dimension = Width
0276     for w in range(0, _nW):
0277         _ChWdFit[m].append([])
```

```
0278         # 3nd dimension = Parameter
0279         for p in range(0, 6):
0280             _ChWdFit[m][w].append([])
0281             # 4nd dimension = Channel Array
0282             _ChWdFit[m][w][p]=array('d')
0283 
0284 # Initialize Width Scan Fitting Data Structure:
0285 _ChAmpFit=[]
0286 # 1st dimension = Measurement
0287 for m in range(0, _nM):
0288     _ChAmpFit.append([])
0289     # 2nd dimension = Amplitude
0290     for a in range(0, _nA):
0291         _ChAmpFit[m].append([])
0292         # 3nd dimension = Parameter
0293         for p in range(0, 6):
0294             _ChAmpFit[m][a].append([])
0295             # 4nd dimension = Channel Array
0296             _ChAmpFit[m][a][p]=array('d')
0297 
0298 ################################################################################
0299 ##                                                                            ##
0300 ##                  Definition of Constant Values to Plot Graphs              ##
0301 ##                                                                            ##
0302 ################################################################################
0303 
0304 ### MULTIGRAPHS:
0305 g1_ = 1.3  #title offset
0306 g2_ = 1    #line color
0307 g3_ = 1    #label font
0308 g4_ = 15   #label size
0309 g5_ = 1    #line width
0310 g6_ = 1    #fill color
0311 g7_ = 1.4  #marker size
0312 g8_ = 20   #marker style
0313 
0314 ## Color Pallet:
0315 RGBcolors=[[255,192,  0],\
0316            [165,165,165],\
0317            [237,125, 49],\
0318            [ 68,114,196],\
0319            [ 91,155,213],\
0320            [112,173, 71],\
0321            [ 38, 68,120]]
0322 
0323 ROOTcolors=[]
0324 for i in range(0, len(RGBcolors)):
0325     _R = RGBcolors[i][0]/255.0
0326     _G = RGBcolors[i][1]/255.0
0327     _B = RGBcolors[i][2]/255.0
0328     ROOTcolors.append(TColor.GetColor(_R,_G,_B))
0329 
0330 ## Markers:
0331 markers=[[ 20, 33, 21, 22, 47, 23, 34],\
0332          [1.4,2.3,1.4,1.7,1.7,1.4,1.7]]
0333 
0334 MultLegX  = ["Programmed Amplitude = 255-aDAC [8-bit value]",\
0335              "Programmed Width = wDAC [binary value]"]
0336 
0337 MultLegY  = ["Adjusted Pulse Area [Vs]",\
0338              "Calculated Pulse Charge [Coulomb]",\
0339              "3mV Trigger Time [s]",\
0340              "Full Width at Half Maximum [s]",\
0341              "Pulse Peak Voltage [V]"]
0342 
0343 BaseTitle = [["Pulse Area vs. Prog. Amplitude",\
0344               "Charge vs. Prog. Amplitude",\
0345               "Timing vs. Prog. Amplitude",\
```

```
0346                "FWHM vs. Prog. Amplitude",\
0347                "Peak vs. Prog. Amplitude"],\
0348               ["Pulse Area vs. Prog. Width",\
0349                "Charge vs. Prog. Width",\
0350                "Timing vs. Prog. Width",\
0351                "FWHM vs. Prog. Width",\
0352                "Peak vs. Prog. Width"]]
0353
0354 stsPos=[[0.8128,0.974124,0.1009,0.3129],\
0355         [0.8128,0.974124,0.3194,0.3969],\
0356         [0.8128,0.974124,0.4033,0.4808],\
0357         [0.8128,0.974124,0.4872,0.5647],\
0358         [0.8128,0.974124,0.5711,0.6486],\
0359         [0.8128,0.974124,0.6550,0.7325],\
0360         [0.8128,0.974124,0.7389,0.8164],\
0361         [0.8128,0.974124,0.8228,0.9003]]
0362
0363 ### HISTOGRAMS:
0364 v1_ = 1.5 #title offset
0365 v2_ = 4   #line color
0366 v3_ = 43  #label font
0367 v4_ = 15  #label size
0368 v5_ = 3   #line width
0369 v6_ = 16  #fill color
0370 v7_ = 1   #marker size
0371 v8_ = 20  #marker style
0372
0373 name3_canvas = "Distribuition of Calibration Parameters per Width Configuration"
0374 HisLegX = [[["p0 = constant parameter [Vs]",\
0375              "p1 = 1st order parameter [Vs/aDAC]",\
0376              "Chi-Square of linear fit"],\
0377             ["p0 = constant parameter [Coloumb]",\
0378              "p1 = 1st order parameter [Coloumb/aDAC]",\
0379              "Chi-Square of linear fit"],\
0380             ["p0 = constant parameter [s]",\
0381              "p1 = 1st order parameter [s/aDAC]",\
0382              "Chi-Square of linear fit"],\
0383             ["p0 = constant parameter [s]",\
0384              "p1 = 1st order parameter [s/aDAC]",\
0385              "Chi-Square of linear fit"],\
0386             ["p0 = constant parameter [V]",\
0387              "p1 = 1st order parameter [V/aDAC]",\
0388              "Chi-Square of linear fit"]],\
0389            [["p0 = constant parameter [Vs]",\
0390              "p1 = 1st order parameter [Vs/wDAC]",\
0391              "Chi-Square of linear fit"],\
0392             ["p0 = constant parameter [Coloumb]",\
0393              "p1 = 1st order parameter [Coloumb/wDAC]",\
0394              "Chi-Square of linear fit"],\
0395             ["p0 = constant parameter [s]",\
0396              "p1 = 1st order parameter [s/wDAC]",\
0397              "Chi-Square of linear fit"],\
0398             ["p0 = constant parameter [s]",\
0399              "p1 = 1st order parameter [s/wDAC]",\
0400              "Chi-Square of linear fit"],\
0401             ["p0 = constant parameter [V]",\
0402              "p1 = 1st order parameter [V/wDAC]",\
0403              "Chi-Square of linear fit"]]]
0404
0405 HisLegY = "channels per bin"
0406
0407 additionalrate = 0.25
0408 diffrate       = 0.25
```

```
0409 q_nbin = 10
0410 ParaName = ["p0","p1","Chi-Sq","Channel","Width","ndf"]
0411
0412 hBaseTitle = [["Area=f(aDAC) Parameter","Charge=f(aDAC) Parameter",\
0413                "Timing=f(aDAC) Parameter","Width=f(aDAC) Parameter",\
0414                "Peak=f(aDAC) Parameter"],\
0415               ["Area=f(wDAC) Parameter","Charge=f(wDAC) Parameter",\
0416               "Timing=f(wDAC) Parameter","Width=f(wDAC) Parameter",\
0417               "Peak=f(wDAC) Parameter"]]
0418
0419 ### Linear Fitting Function:
0420 func = TF1("func", "[0] + [1]*x")
0421
0422 ################################################################################
0423 ##                                                                            ##
0424 ##                  Getting Fit Parameters and Building Graphs                ##
0425 ##                                                                            ##
0426 ################################################################################
0427
0428 ############################## Parameter versus wDAC
##############################
0429
0430 LogIt("---- GETTING FIT PARAMETERS FOR wDAC ----- ")
0431
0432 ChWdMultiGfx = []
0433 ChWdMultiLeg = []
0434 grChWd = []
0435 for m in range(0, _nM):
0436     ChWdMultiGfx.append([])
0437     ChWdMultiLeg.append([])
0438     grChWd.append([])
0439     for w in range(0, _nW):
0440         grChWd[m].append([])
0441         for c in range(0, _nC):
0442             start = (w*_nA)+c*(_nW*_nA)
0443             stop = (w*_nA)+c*(_nW*_nA)+_nA
0444             delta = stop-start
0445             # Test channel and amplitude of data
0446             testOpCh = _Data[0][start]
0447             for value in _Data[0][start:stop]:
0448                 if value != testOpCh:
0449                     LogIt("WARN - Inconsistent Channel
Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0450                           % (_Data[0][start],_Data[1][start],255-_Data[2][start]))
0451             testOpWd = w+min(_Data[1])
0452             for value in _Data[1][start:stop]:
0453                 if value != testOpWd:
0454                     LogIt("WARN: Inconsistent wDAC Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0455                           % (_Data[0][start],_Data[1][start],255-_Data[2][start]))
0456             # Build Output Channel INDEX:
0457             if w == 0:
0458                 ChWdMultiGfx[m].append([])
0459                 ChWdMultiGfx[m][c] = TMultiGraph()
0460                 ChWdMultiLeg[m].append([])
0461                 ChWdMultiLeg[m][c] = \
0462                     TLegend(stsPos[0][0],stsPos[0][2],stsPos[0][1],stsPos[0][3])
0463                 ChWdMultiLeg[m][c].SetFillColor(0);
0464                 ChWdMultiLeg[m][c].SetFillStyle(1001);
0465                 if m == 0: nChMultiGfx.append(testOpCh)
0466             # Build Output Width INDEX:
0467             if c == 0:
0468                 if m == 0: wDACMultiGfx.append(testOpWd)
0469             grChWd[m][w].append([])
```

```
0470            kSta = start
0471            if m == 0:
0472                for a in range(0, delta):
0473                    if _Data[4][kSta+a] == 0.0:
0474                        start = kSta+a+1
0475                        LogIt("INFO: Null Area Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0476                        %(_Data[0][kSta+a],_Data[1][kSta+a],255-_Data[2][kSta+a]))
0477                grChWd[m][w][c] = TGraphErrors(len(_Data[2][start:stop]),\
0478                                  _Data[2][start:stop],  _Data[4][start:stop],\
0479                                  _Data[3][start:stop], _Data[5][start:stop])
0480            if m == 1:
0481                for a in range(0, delta):
0482                    if _Data[6][kSta+a] == 0.0:
0483                        start = kSta+a+1
0484                        LogIt("INFO: Null Charg Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0485                        % (_Data[0][kSta+a],_Data[1][kSta+a],255-_Data[2][kSta+a]))
0486                grChWd[m][w][c] = TGraphErrors(len(_Data[2][start:stop]),\
0487                                  _Data[2][start:stop],  _Data[6][start:stop],\
0488                                  _Data[3][start:stop], _Data[7][start:stop])
0489            if m == 2:
0490                for a in range(0, delta):
0491                    if _Data[12][kSta+a] == 0.0:
0492                        start = kSta+a+1
0493                        LogIt("INFO: Null Timin Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0494                        % (_Data[0][kSta+a],_Data[1][kSta+a],255-_Data[2][kSta+a]))
0495                grChWd[m][w][c] = TGraphErrors(len(_Data[2][start:stop]),\
0496                                  _Data[2][start:stop],  _Data[12][start:stop],\
0497                                  _Data[3][start:stop], _Data[13][start:stop])
0498            if m == 3:
0499                for a in range(0, delta):
0500                    if _Data[8][kSta+a] == 0.0:
0501                        start = kSta+a+1
0502                        LogIt("INFO: Null Width Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0503                        % (_Data[0][kSta+a],_Data[1][kSta+a],255-_Data[2][kSta+a]))
0504                grChWd[m][w][c] = TGraphErrors(len(_Data[2][start:stop]),\
0505                                  _Data[2][start:stop],  _Data[8][start:stop],\
0506                                  _Data[3][start:stop], _Data[9][start:stop])
0507            if m == 4:
0508                for a in range(0, delta):
0509                    if _Data[10][kSta+a] == 0.0:
0510                        start = kSta+a+1
0511                        LogIt("INFO: Null Peak Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0512                        % (_Data[0][kSta+a],_Data[1][kSta+a],255-_Data[2][kSta+a]))
0513                grChWd[m][w][c] = TGraphErrors(len(_Data[2][start:stop]),\
0514                                  _Data[2][start:stop],  _Data[10][start:stop],\
0515                                  _Data[3][start:stop], _Data[11][start:stop])
0516            grChWd[m][w][c].Fit("func","Q")
0517            grChWd[m][w][c].SetLineColor(ROOTcolors[w])
0518            grChWd[m][w][c].SetLineWidth( g5_ )
0519            grChWd[m][w][c].SetMarkerColor(ROOTcolors[w])
0520            grChWd[m][w][c].SetMarkerStyle(markers[0][w])
0521            grChWd[m][w][c].SetMarkerSize(markers[1][w])
0522            ChWdMultiGfx[m][c].Add(grChWd[m][w][c])
0523
0524            _a   = func.GetParameter(0)
0525            _b   = func.GetParameter(1)
0526            _x   = func.GetChisquare()
0527            _Ch  = testOpCh
0528            _Wd  = testOpWd
0529            _ndf = func.GetNDF()
0530
0531            _ChWdFit[m][w][0].append(_a)
0532            _ChWdFit[m][w][1].append(_b)
0533            _ChWdFit[m][w][2].append(_x)
0534            _ChWdFit[m][w][3].append(_Ch)
0535            _ChWdFit[m][w][4].append(_Wd)
0536            _ChWdFit[m][w][5].append(_ndf)
0537
0538 for m in range(0, _nM):
0539     for c in range(0, _nC):
```

```
0540         for w in range(_nW-1, -1, -1):
0541             ChWdMultiLeg[m][c].AddEntry(grChWd[m][w][c],"Width 0x%02X"\
0542                                                 % (wDACMultiGfx[w]),"lp")
0543 
0544 ### Compute worst error:
0545 _ChWdWorstErr = []
0546 for m in range(0, _nM):
0547     _ChWdWorstErr.append([])
0548     for w in range(0, _nW):
0549         _ChWdWorstErr[m].append([])
0550         _ChWdWorstErr[m][w] = 0
0551         for c in range(0, _nC):
0552             ChiByNdf = _ChWdFit[m][w][2][c] / _ChWdFit[m][w][5][c]
0553             if ChiByNdf > _ChWdWorstErr[m][w]: _ChWdWorstErr[m][w] = ChiByNdf
0554 
0555 ############################## Parameter versus aDAC
##############################
0556 
0557 LogIt("---- GETTING FIT PARAMETERS FOR aDAC ----- ")
0558 
0559 ChAmpMultiGfx = []
0560 ChAmpMultiLeg = []
0561 rdChAmp = []
0562 grChAmp = []
0563 for m in range(0, _nM):
0564     ChAmpMultiGfx.append([])
0565     ChAmpMultiLeg.append([])
0566     grChAmp.append([])
0567     for a in range(0, _nA):
0568         rdChAmp.append([])
0569         grChAmp[m].append([])
0570         for c in range(0, _nC):
0571             rdChAmp[a].append([])
0572             if m == 0:
0573                 for j in range(0, 14):
0574                     tempData=array('d')
0575                     for i in range(0, len(_Data[2])):
0576                         if _CountSet[2][a][0] == _Data[2][i] and\
0577                            _CountSet[0][c][0] == _Data[0][i]:
0578                             tempData.append(_Data[j][i])
0579                     rdChAmp[a][c].append(tempData)
0580                     #rdChAmp[a][c][d][w]
0581 
0582             grChAmp[m][a].append([])
0583             start = 0
0584             stop  = _nW
0585             # Test channel and amplitude of data
0586             testOpAmp = rdChAmp[a][c][2][start]
0587             for value in rdChAmp[a][c][2]:
0588                 if value != testOpAmp:
0589                     LogIt("WARN: Inconsistent aDAC -
Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0590                     % (rdChAmp[a][c][0][start], rdChAmp[a][c][1][start],\
0591                     255-rdChAmp[a][c][2][start]))
0592             # Build Output Amp INDEX:
0593             if c == 0:
0594                 if m == 0: aDACMultiGfx.append(255-testOpAmp)
0595 
0596             # Build Legend Object:
0597             if a == 0:
0598                 ChAmpMultiGfx[m].append([])
0599                 ChAmpMultiGfx[m][c] = TMultiGraph()
0600                 ChAmpMultiLeg[m].append([])
0601                 ChAmpMultiLeg[m][c] = \
0602                     TLegend(stsPos[0][0],stsPos[0][2],stsPos[0][1],stsPos[0][3])
0603                 ##ChAmpMultiLeg[m][c].SetTextFont(32);
```

```
0604                ChAmpMultiLeg[m][c].SetFillColor(0);
0605                ChAmpMultiLeg[m][c].SetFillStyle(1001);
0606            kSta = start
0607            if m == 0:
0608                for w in range(0, _nW):
0609                    if rdChAmp[a][c][4][kSta+w] == 0.0:
0610                        start = kSta+w+1
0611                        LogIt("INFO: Null Area Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0612                        % (rdChAmp[a][c][0][kSta+w],rdChAmp[a][c][1][kSta+w],\
0613                        255-rdChAmp[a][c][2][kSta+w]))
0614                grChAmp[m][a][c] = TGraphErrors(len(rdChAmp[a][c][2][start:stop]),\
0615                    rdChAmp[a][c][1][start:stop],  rdChAmp[a][c][4][start:stop],\
0616                    rdChAmp[a][c][3][start:stop], rdChAmp[a][c][5][start:stop])
0617            if m == 1:
0618                for w in range(0, _nW):
0619                    if rdChAmp[a][c][6][kSta+w] == 0.0:
0620                        start = kSta+w+1
0621                        LogIt("INFO: Null Charg Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0622                        % (rdChAmp[a][c][0][kSta+w],rdChAmp[a][c][1][kSta+w],\
0623                        255-rdChAmp[a][c][2][kSta+w]))
0624                grChAmp[m][a][c] = TGraphErrors(len(rdChAmp[a][c][2][start:stop]),\
0625                    rdChAmp[a][c][1][start:stop],  rdChAmp[a][c][6][start:stop],\
0626                    rdChAmp[a][c][3][start:stop], rdChAmp[a][c][7][start:stop])
0627            if m == 2:
0628                for w in range(0, _nW):
0629                    if rdChAmp[a][c][12][kSta+w] == 0.0:
0630                        start = kSta+w+1
0631                        LogIt("INFO: Null Timin Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0632                        % (rdChAmp[a][c][0][kSta+w],rdChAmp[a][c][1][kSta+w],\
0633                        255-rdChAmp[a][c][2][kSta+w]))
0634                grChAmp[m][a][c] = TGraphErrors(len(rdChAmp[a][c][2][start:stop]),\
0635                    rdChAmp[a][c][1][start:stop],  rdChAmp[a][c][12][start:stop],
0636                    rdChAmp[a][c][3][start:stop], rdChAmp[a][c][13][start:stop])
0637            if m == 3:
0638                for w in range(0, _nW):
0639                    if rdChAmp[a][c][8][kSta+w] == 0.0:
0640                        start = kSta+w+1
0641                        LogIt("INFO: Null Width Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0642                        % (rdChAmp[a][c][0][kSta+w],rdChAmp[a][c][1][kSta+w],\
0643                        255-rdChAmp[a][c][2][kSta+w]))
0644                grChAmp[m][a][c] = TGraphErrors(len(rdChAmp[a][c][2][start:stop]),\
0645                    rdChAmp[a][c][1][start:stop],  rdChAmp[a][c][8][start:stop],\
0646                    rdChAmp[a][c][3][start:stop], rdChAmp[a][c][9][start:stop])
0647            if m == 4:
0648                for w in range(0, _nW):
0649                    if rdChAmp[a][c][10][kSta+w] == 0.0:
0650                        start = kSta+w+1
0651                        LogIt("INFO: Null Peak Skip Ch0x%02X_Wid0x%02X_Amp0x%02X"\
0652                        % (rdChAmp[a][c][0][kSta+w],rdChAmp[a][c][1][kSta+w],\
0653                        255-rdChAmp[a][c][2][kSta+w]))
0654                grChAmp[m][a][c] = TGraphErrors(len(rdChAmp[a][c][2][start:stop]),\
0655                    rdChAmp[a][c][1][start:stop],  rdChAmp[a][c][10][start:stop],\
0656                    rdChAmp[a][c][3][start:stop], rdChAmp[a][c][11][start:stop])
0657            grChAmp[m][a][c].Fit("func","Q")
0658            sMarkColor  =   1 # Default Color is Black
0659            sMarkStyle  =  21 # Default Style is a Square
0660            sMarkSize   = 0.8 # Default Size is Small
0661            if a == 0:
0662                sMarkColor = ROOTcolors[0]
0663                sMarkStyle = markers[0][0]
0664                sMarkSize  = markers[1][0]
0665            if a == 3:
0666                sMarkColor = ROOTcolors[1]
0667                sMarkStyle = markers[0][1]
0668                sMarkSize  = markers[1][1]
0669            if a == 6:
0670                sMarkColor = ROOTcolors[2]
0671                sMarkStyle = markers[0][2]
0672                sMarkSize  = markers[1][2]
0673            if a == 9:
```

```
0674                sMarkColor = ROOTcolors[3]
0675                sMarkStyle = markers[0][3]
0676                sMarkSize  = markers[1][3]
0677            if a == 11:
0678                sMarkColor = ROOTcolors[4]
0679                sMarkStyle = markers[0][4]
0680                sMarkSize  = markers[1][4]
0681            if a == 13:
0682                sMarkColor = ROOTcolors[5]
0683                sMarkStyle = markers[0][5]
0684                sMarkSize  = markers[1][5]
0685            if a == 15:
0686                sMarkColor = ROOTcolors[6]
0687                sMarkStyle = markers[0][6]
0688                sMarkSize  = markers[1][6]
0689            grChAmp[m][a][c].SetLineColor(sMarkColor)
0690            grChAmp[m][a][c].SetLineWidth( g5_ )
0691            grChAmp[m][a][c].SetMarkerColor(sMarkColor)
0692            grChAmp[m][a][c].SetMarkerStyle(sMarkStyle)
0693            grChAmp[m][a][c].SetMarkerSize(sMarkSize)
0694            ChAmpMultiGfx[m][c].Add(grChAmp[m][a][c])
0695
0696            _a   = func.GetParameter(0)
0697            _b   = func.GetParameter(1)
0698            _x   = func.GetChisquare()
0699            _Ch  = testOpCh
0700            _Wd  = testOpWd
0701            _ndf = func.GetNDF()
0702
0703            _ChAmpFit[m][a][0].append(_a)
0704            _ChAmpFit[m][a][1].append(_b)
0705            _ChAmpFit[m][a][2].append(_x)
0706            _ChAmpFit[m][a][3].append(_Ch)
0707            _ChAmpFit[m][a][4].append(_Wd)
0708            _ChAmpFit[m][a][5].append(_ndf)
0709
0710 for m in range(0, _nM):
0711     for c in range(0, _nC):
0712         for a in range(_nA-1, -1, -1):
0713             if a == 0 or a == 3 or a == 6 or a == 9 or a == 11 or a == 13 or a ==
15:
0714                 ChAmpMultiLeg[m][c].AddEntry(grChAmp[m][a][c],"Amplitude 0x%02X"\
0715                                                                 % (aDACMul-
tiGfx[a]),"lp")
0716
0717 ### Compute worst error:
0718 _ChAmpWorstErr = []
0719 for m in range(0, _nM):
0720     _ChAmpWorstErr.append([])
0721     for a in range(0, _nA):
0722         _ChAmpWorstErr[m].append([])
0723         _ChAmpWorstErr[m][a] = 0
0724         for c in range(0, _nC):
0725             ChiByNdf = _ChAmpFit[m][a][2][c] / _ChAmpFit[m][a][5][c]
0726             if ChiByNdf > _ChAmpWorstErr[m][a]: _ChAmpWorstErr[m][a] = ChiByNdf
0727
0728 ### Y-Axis Maximum Adjust:
0729 YaxisClearance = 1.05
0730 GfxMax = [YaxisClearance*max(_Data[4]),\
0731           YaxisClearance*max(_Data[6]),\
0732           YaxisClearance*max(_Data[12]),\
0733           YaxisClearance*max(_Data[8]),\
0734           YaxisClearance*max(_Data[10])]
0735
0736 ##############################################################################
0737 ##                                                                          ##
0738 ##         Getting parameters from fitted charge vs. amplitude functions    ##
0739 ##                                                                          ##
0740 ##############################################################################
```

```
0741
0742 ############################ Parameter versus wDAC
#############################
0743
0744 LogIt("---- GENERATING AMPLITUDE SCAN PLOTS ----- ")
0745
0746 PlotCount = 0
0747
0748 ChWdMultiGfxCanvas = []
0749 ChWdSts=[]
0750 _mC = min(PlotChLimit, _nC)
0751 for m in range(0, _nM):
0752     ChWdMultiGfxCanvas.append([])
0753     ChWdSts.append([])
0754     for c in range(0, _mC):
0755         ChWdMultiGfxCanvas[m].append([])
0756         ChWdSts[m].append([])
0757         n = c
0758         if GfxSingleCanvas: n = 0
0759         ChWdMultiGfxCanvas[m][n] = TCanvas("m%i_c0x%02X" % (m, nChMultiGfx[c]),\
0760                                    "Measurement vs. aDAC for various
wDAC",1370,800)
0761         GfxTitle = "%s @ Channel 0x%02X" % (BaseTitle[0][m], nChMultiGfx[c])
0762         ChWdMultiGfx[m][c].SetTitle(GfxTitle)
0763         ChWdMultiGfx[m][c].Draw("AP")
0764         ChWdMultiLeg[m][c].Draw()
0765         ChWdMultiGfxCanvas[m][n].Update()
0766
0767         for w in range(0, _nW):
0768             ChWdSts[m][c].append([])
0769             ChWdStsName  = "ChWd_m%ic0x%02Xw0x%02X"\
0770                            % (m,nChMultiGfx[c],wDACMultiGfx[w])
0771             _StatsBox = grChWd[m][w][c].FindObject("stats")
0772             _StatsBox.SetName(ChWdStsName)
0773             _StatsBox.SetX1NDC(stsPos[w+1][0])
0774             _StatsBox.SetX2NDC(stsPos[w+1][1])
0775             _StatsBox.SetY1NDC(stsPos[w+1][2])
0776             _StatsBox.SetY2NDC(stsPos[w+1][3])
0777             _StatsBox.SetTextSize(0.0207)
0778             _StatsBox.SetLineWidth(1)
0779             _StatsBox.SetLineColor(ROOTcolors[w])
0780
0781             fitLine = grChWd[m][w][c].FindObject("func")
0782             fitLine.SetLineColor(910)
0783             fitLine.SetLineWidth(2)
0784             fitLine.SetLineStyle(2)
0785             fitLine.SetRange(0,255)
0786             fitLine.SetName("f_%s" % (ChWdStsName))
0787
0788         x = ChWdMultiGfx[m][c].GetXaxis()
0789         x.SetTitleOffset( g1_)
0790         x.SetTitle(MultLegX[0])
0791         y = ChWdMultiGfx[m][c].GetYaxis()
0792         y.SetTitleOffset( g1_)
0793         y.SetTitle(MultLegY[m])
0794         if m != 2: ChWdMultiGfx[m][c].SetMinimum(0.0)
0795         if m != 2: ChWdMultiGfx[m][c].SetMaximum(GfxMax[m])
0796         gPad.SetRightMargin(0.2)
0797         gPad.Modified()
0798
0799         ChWdMultiGfxCanvas[m][n].Update()
0800
0801         GfxSave_PdfBookmark = "Title:%s_Ch0x%02X" % (MeasNames[m], nChMultiGfx[c])
0802         GfxSave_PdfFileName = "%s/AmpScan_%s.pdf" % (PdfDir, MeasNames[m])
0803         if c ==     0:
0804             GfxSave_PdfFileName = "%s/AmpScan_%s.pdf(" % (PdfDir, MeasNames[m])
0805         if c == _mC-1:
```

```
0806             GfxSave_PdfFileName = "%s/AmpScan_%s.pdf)" % (PdfDir, MeasNames[m])
0807         if ExportPdf:
0808             ChWdMultiGfxCanvas[m][n].Print(GfxSave_PdfFileName,GfxSave_PdfBookmark)
0809         GfxSave_EpsFileName = "%s/%s/AmpScan/%s_c0x%02X.eps"\
0810                         % (EpsDirectory, MeasNames[m], MeasNames[m], nChMul-
tiGfx[c])
0811         if ExportEps:
0812             ChWdMultiGfxCanvas[m][n].SaveAs(GfxSave_EpsFileName)
0813
0814         PlotCount += 1
0815
0816 LogIt("INFO - %4s plots generated" % PlotCount)
0817
0818 ############################## Parameter versus aDAC
##############################
0819
0820 LogIt("------ GENERATING WIDTH SCAN PLOTS ------- ")
0821
0822 PlotCount = 0
0823
0824 ChAmpMultiGfxCanvas = []
0825 ChAmpSts=[]
0826 _mC = min(PlotChLimit, _nC)
0827 for m in range(0, _nM):
0828     ChAmpMultiGfxCanvas.append([])
0829     ChAmpSts.append([])
0830     for c in range(0, _mC):
0831         ChAmpMultiGfxCanvas[m].append([])
0832         ChAmpSts[m].append([])
0833         n = c
0834         if GfxSingleCanvas: n = 0
0835         ChAmpMultiGfxCanvas[m][n] = TCanvas("ChAmp_m%i_c0x%02X"\
0836              % (m, nChMultiGfx[c]),"Measurement vs wDAC for various
aDAC",1370,800)
0837         GfxTitle = "%s @ Channel 0x%02X" % (BaseTitle[1][m], nChMultiGfx[c])
0838         ChAmpMultiGfx[m][c].SetTitle(GfxTitle)
0839         ChAmpMultiGfx[m][c].Draw("AP")
0840         ChAmpMultiLeg[m][c].Draw()
0841         ChAmpMultiGfxCanvas[m][n].Update()
0842
0843         for a in range(0, _nA):
0844             ChAmpSts[m][c].append([])
0845             ChAmpStsName  = "ChAmp_m%ic0x%02Xa0x%02X"\
0846                             % (m,nChMultiGfx[c], aDACMultiGfx[a])
0847             _StatsBox = grChAmp[m][a][c].FindObject("stats")
0848             _StatsBox.SetName(ChAmpStsName)
0849             modA = 0
0850             if a > 2: modA = 1
0851             if a > 5: modA = 2
0852             if a > 7: modA = 3
0853             if a > 9: modA = 4
0854             if a > 11: modA = 5
0855             if a > 13: modA = 6
0856             _StatsBox.SetX1NDC(stsPos[modA+1][0])
0857             _StatsBox.SetX2NDC(stsPos[modA+1][1])
0858             _StatsBox.SetY1NDC(stsPos[modA+1][2])
0859             _StatsBox.SetY2NDC(stsPos[modA+1][3])
0860             _StatsBox.SetTextSize(0.0207)
0861             _StatsBox.SetLineWidth(1)
0862             _StatsBox.SetLineColor(ROOTcolors[modA])
0863
0864             fitLine = grChAmp[m][a][c].FindObject("func")
0865             fLineColor =   1 # Default Color is Black
0866             fLineStyle =   1 # Default Style is Continuous
0867             fLineWidth =   1 # Default Width is 1 Pixel
0868             if a==0 or a==3 or a==6 or a==9 or a==11 or a==13 or a==15:
0869                 fLineColor = 910 # Selected Color is Pink
```

```
0870                 fLineStyle =   2 # Selected Style is Doted
0871                 fLineWidth =   2 # Selected Width is 2 Pixels
0872             fitLine.SetLineColor(fLineColor)
0873             fitLine.SetLineWidth(fLineWidth)
0874             fitLine.SetLineStyle(fLineStyle)
0875             fitLine.SetRange(min(rdChAmp[a][c][1]),max(rdChAmp[a][c][1]))
0876             fitLine.SetName("f_%s" % (ChAmpStsName))
0877
0878         x = ChAmpMultiGfx[m][c].GetXaxis()
0879         x.SetTitleOffset( g1_)
0880         x.SetTitle(MultLegX[1])
0881         y = ChAmpMultiGfx[m][c].GetYaxis()
0882         y.SetTitleOffset( g1_)
0883         y.SetTitle(MultLegY[m])
0884         if m != 2: ChAmpMultiGfx[m][c].SetMinimum(0.0)
0885         if m != 2: ChAmpMultiGfx[m][c].SetMaximum(GfxMax[m])
0886         gPad.SetRightMargin(0.2)
0887         gPad.Modified()
0888
0889         ChAmpMultiGfxCanvas[m][n].Update()
0890
0891         GfxSave_PdfBookmark = "Title:%s_Ch0x%02X" % (MeasNames[m], nChMultiGfx[c])
0892         GfxSave_PdfFileName = "%s/WidScan_%s.pdf" % (PdfDir, MeasNames[m])
0893         if c ==     0:
0894             GfxSave_PdfFileName = "%s/WidScan_%s.pdf(" % (PdfDir, MeasNames[m])
0895         if c == _mC-1:
0896             GfxSave_PdfFileName = "%s/WidScan_%s.pdf)" % (PdfDir, MeasNames[m])
0897         if ExportPdf:
0898             ChAmpMultiGfxCanvas[m][n].Print(GfxSave_PdfFileName,GfxSave_PdfBook-
mark)
0899         GfxSave_EpsFileName = "%s/%s/WidScan/%s_c0x%02X.eps"\
0900                         % (EpsDirectory, MeasNames[m], MeasNames[m], nChMul-
tiGfx[c])
0901         if ExportEps:
0902             ChAmpMultiGfxCanvas[m][n].SaveAs(GfxSave_EpsFileName)
0903
0904         PlotCount += 1
0905
0906 LogIt("INFO - %4s plots generated" % PlotCount)
0907
0908 ##############################################################################
0909 ##                                                                          ##
0910 ##                 Generating Histograms from fitting parameters            ##
0911 ##                                                                          ##
0912 ##############################################################################
0913
0914 ############################## Parameter versus aDAC
##############################
0915
0916 LogIt("-- GENERATING AMPLITUDE SCAN HISTOGRAMS -- ")
0917
0918 PlotCount = 0
0919
0920 ChWdHisMinMax = []
0921 for m in range(0, _nM):
0922     ChWdHisMinMax.append([])
0923     for w in range(0, _nW):
0924         ChWdHisMinMax[m].append([])
0925         for p in range(0, 3):
0926             ChWdHisMinMax[m][w].append([])
0927             ChWdHisMinMax[m][w][p]=array('d')
0928             DeltaQ                = max(_ChWdFit[m][w][p]) - min(_ChWdFit[m][w][p])
0929             ChWdHisMinMax[m][w][p].append(min(_ChWdFit[m][w][p])\
0930                                           - DeltaQ*additionalrate)
0931             ChWdHisMinMax[m][w][p].append(max(_ChWdFit[m][w][p])\
0932                                           + DeltaQ*(additionalrate+diffrate))
0933
0934 hChWd = []
0935 _ChWdFitPar=[]
```

```
0936 for m in range(0, _nM):
0937     hChWd.append([])
0938     _ChWdFitPar.append([])
0939     for w in range(0, _nW):
0940         hChWd[m].append([])
0941         _ChWdFitPar[m].append([])
0942         for p in range(0, 3):
0943             hChWd[m][w].append([])
0944             _ChWdFitPar[m][w].append([])
0945             hChWd[m][w][p] = TH1F("h%i_W0x%02X_%s"\
0946                                   % (m, wDACMultiGfx[w],ParaName[p]),"",q_nbin,\
0947                                   ChWdHisMinMax[m][w][p][0],ChWdHisMinMax[m][w][p][1])
0948             for value in _ChWdFit[m][w][p]:
0949                 hChWd[m][w][p].Fill(value)
0950             hChWd[m][w][p].Fit("gaus","Q")
0951             fitLine = hChWd[m][w][p].FindObject("gaus")
0952             fitLine.SetName("m%iw0x%02X%s" % (m,wDACMultiGfx[w],ParaName[p]))
0953             _Const = fitLine.GetParameter(0)
0954             _Mean  = fitLine.GetParameter(1)
0955             _Sigma = fitLine.GetParameter(2)
0956 
0957             _ChWdFitPar[m][w][p].append(_Const)
0958             _ChWdFitPar[m][w][p].append(_Mean)
0959             _ChWdFitPar[m][w][p].append(_Sigma)
0960 
0961 for m in range(0, _nM):
0962     for w in range(0, _nW):
0963         for p in range(0, 3):
0964             GfxTitle = "%s %s @ wDAC=0x%02X"\
0965                        % (hBaseTitle[0][m], ParaName[p], wDACMultiGfx[w])
0966             x = hChWd[m][w][p].GetXaxis()
0967             x.SetTitleOffset(v1_)
0968             x.SetLabelFont(v3_)
0969             x.SetLabelSize(v4_)
0970             y = hChWd[m][w][p].GetYaxis()
0971             y.SetTitleOffset(v1_)
0972             y.SetLabelFont(v3_)
0973             y.SetLabelSize(v4_)
0974             hChWd[m][w][p].SetLineColor(v2_)
0975             hChWd[m][w][p].SetLineWidth(v5_)
0976             hChWd[m][w][p].SetFillColor(v6_)
0977             hChWd[m][w][p].SetMarkerSize(v7_)
0978             hChWd[m][w][p].SetMarkerStyle(v8_)
0979             hChWd[m][w][p].SetTitle(GfxTitle)
0980             hChWd[m][w][p].SetXTitle(HisLegX[0][m][p])
0981             hChWd[m][w][p].SetYTitle(HisLegY)
0982 
0983 _ChWdFitCanvas=[]
0984 _mW = min(HistLimit, _nW)
0985 for m in range(0, _nM):
0986     _ChWdFitCanvas.append([])
0987     for w in range(0, _mW):
0988         _ChWdFitCanvas[m].append([])
0989         n = w
0990         if HisSingleCanvas: n = 0
0991         _ChWdFitCanvas[m][n] = TCanvas("ChWd_m%i_w0x%02X"\
0992                                        % (m, wDACMultiGfx[w]),name3_can-
vas,1395,450)
0993         _ChWdFitCanvas[m][n].Divide(3,1)
0994         for p in range(0, 3):
0995             _ChWdFitCanvas[m][n].cd(p+1)
0996             hChWd[m][w][p].Draw()
0997 
0998         _ChWdFitCanvas[m][n].Update()
0999 
```

```
1000         GfxSave_PdfBookmark = "Title:%s_Wid0x%02X"%(MeasNames[m], wDACMul-
tiGfx[w])
1001         GfxSave_PdfFileName = "%s/%sAmpScanFitHist.pdf" % (PdfDir, MeasNames[m])
1002         if w ==     0:
1003             GfxSave_PdfFileName = "%s/%sAmpScanFitHist.pdf("%(PdfDir, MeasNames[m])
1004         if w == _nW-1:
1005             GfxSave_PdfFileName = "%s/%sAmpScanFitHist.pdf)"%(PdfDir, MeasNames[m])
1006         if ExportPdf:
1007             _ChWdFitCanvas[m][n].Print(GfxSave_PdfFileName,GfxSave_PdfBookmark)
1008         GfxSave_EpsFileName = "%s/%s/_%s_AmpScanFitHist_W0x%02X.eps"\
1009                         % (EpsDirectory, MeasNames[m], MeasNames[m], wDACMul-
tiGfx[w])
1010         if ExportEps:
1011             _ChWdFitCanvas[m][n].SaveAs(GfxSave_EpsFileName)
1012
1013         PlotCount += 1
1014
1015 LogIt("INFO - %4s histograms generated" % PlotCount)
1016
1017 ### Generate Tables:
1018
1019 LogIt("----- DRAWING AMPLITUDE SCAN TABLES ------ ")
1020
1021 PlotCount = 0
1022
1023 ChWdParTableLabel = ["Area = p0 + p1*[aDAC]",\
1024                      "Charge = p0 + p1*[aDAC]",\
1025                      "Timing = p0 + p1*[aDAC]",\
1026                      "Width = p0 + p1*[aDAC]",\
1027                      "Peak = p0 + p1*[aDAC]"]
1028
1029 TableUnit = [["[binary]","[Volt*second]","[V*s / aDAC]","X^2/ndf"],\
1030              ["[binary]","[Coulomb]","[Coulomb / aDAC]","X^2/ndf"],\
1031              ["[binary]","[second]","[s / aDAC]","X^2/ndf"],\
1032              ["[binary]","[second]","[s / aDAC]","X^2/ndf"],\
1033              ["[binary]","[Volt]","[V / aDAC]","X^2/ndf"]]
1034
1035 _ChWdErrorColor=[]
1036 for m in range(0, _nM):
1037     _ChWdErrorColor.append([])
1038     for w in range(0, _nW):
1039         _ChWdErrorColor[m].append([])
1040         _ChWdErrorColor[m][w] = 1
1041         if _ChWdWorstErr[m][w] > 1: _ChWdErrorColor[m][w] = 874
1042         if _ChWdWorstErr[m][w] > 2: _ChWdErrorColor[m][w] = 2
1043
1044 ChWdParTableCanvas=[]
1045 ChWdParTableTitle=[]
1046 ChWdParTableCol1=[]
1047 ChWdParTableCol2=[]
1048 ChWdParTableCol3=[]
1049 ChWdParTableCol4=[]
1050
1051 for m in range(0, _nM):
1052     ChWdParTableCanvas.append([])
1053     ChWdParTableTitle.append([])
1054     ChWdParTableCol1.append([])
1055     ChWdParTableCol2.append([])
1056     ChWdParTableCol3.append([])
1057     ChWdParTableCol4.append([])
1058     ChWdParTableCanvas[m] = TCanvas("AmpScanParTable_m%s" % (m),\
```

```
1059                             "%s Table for f(aDAC)" % MeasNames[m], 700, 405)
1060
1061      ChWdParTableTitle[m] = TPaveText(.25,.849,.75,.9698)
1062      ChWdParTableTitle[m].AddText(ChWdParTableLabel[m])
1063      ChWdParTableTitle[m].Draw()
1064
1065      ChWdParTableCol1[m] = TPaveText(.02,.07,.155,.81)
1066      ChWdParTableCol1[m].AddText("wDAC")
1067      ChWdParTableCol1[m].AddLine(.0,(_nW+1)*1./(_nW+2),1.,(_nW+1)*1./(_nW+2))
1068      ChWdParTableCol1[m].AddText(TableUnit[m][0])
1069      ChWdParTableCol1[m].AddLine(.0,(_nW-0)*1./(_nW+2),1.,(_nW-0)*1./(_nW+2))
1070      for w in range(0, _nW):
1071          lastrow = ChWdParTableCol1[m].AddText("0x%02X" % (wDACMultiGfx[w]))
1072          lastrow.SetTextColor(_ChWdErrorColor[m][w])
1073      ChWdParTableCol1[m].SetTextSize(0.05785)
1074      ChWdParTableCol1[m].Draw()
1075
1076      ChWdParTableCol2[m] = TPaveText(.155,.07,.489,.81)
1077      ChWdParTableCol2[m].AddText("p0 Mean #pm Sigma")
1078      ChWdParTableCol2[m].AddLine(.0,(_nW+1)*1./(_nW+2),1.,(_nW+1)*1./(_nW+2))
1079      ChWdParTableCol2[m].AddText(TableUnit[m][1])
1080      ChWdParTableCol2[m].AddLine(.0,(_nW-0)*1./(_nW+2),1.,(_nW-0)*1./(_nW+2))
1081      for w in range(0, _nW):
1082          lastrow = ChWdParTableCol2[m].AddText("%.3E #pm %.3E"\
1083                                %(_ChWdFitPar[m][w][0][1], _ChWdFitPar[m][w][0][2]))
1084          lastrow.SetTextColor(_ChWdErrorColor[m][w])
1085      ChWdParTableCol2[m].SetTextSize(0.05785)
1086      ChWdParTableCol2[m].Draw()
1087
1088      ChWdParTableCol3[m] = TPaveText(.489,.07,.823,.81)
1089      ChWdParTableCol3[m].AddText("p1 Mean #pm Sigma")
1090      ChWdParTableCol3[m].AddLine(.0,(_nW+1)*1./(_nW+2),1.,(_nW+1)*1./(_nW+2))
1091      ChWdParTableCol3[m].AddText(TableUnit[m][2])
1092      ChWdParTableCol3[m].AddLine(.0,(_nW-0)*1./(_nW+2),1.,(_nW-0)*1./(_nW+2))
1093      for w in range(0, _nW):
1094          lastrow = ChWdParTableCol3[m].AddText("%.3E #pm %.3E"\
1095                                %(_ChWdFitPar[m][w][1][1], _ChWdFitPar[m][w][1][2]))
1096          lastrow.SetTextColor(_ChWdErrorColor[m][w])
1097      ChWdParTableCol3[m].SetTextSize(0.05785)
1098      ChWdParTableCol3[m].Draw()
1099
1100      ChWdParTableCol4[m] = TPaveText(.823,.07,.98,.81)
1101      ChWdParTableCol4[m].AddText("Max_Err")
1102      ChWdParTableCol4[m].AddLine(.0,(_nW+1)*1./(_nW+2),1.,(_nW+1)*1./(_nW+2))
1103      ChWdParTableCol4[m].AddText(TableUnit[m][3])
1104      ChWdParTableCol4[m].AddLine(.0,(_nW-0)*1./(_nW+2),1.,(_nW-0)*1./(_nW+2))
1105      for w in range(0, _nW):
1106          lastrow = ChWdParTableCol4[m].AddText("%.3E"%(_ChWdWorstErr[m][w]))
1107          lastrow.SetTextColor(_ChWdErrorColor[m][w])
1108      ChWdParTableCol4[m].SetTextSize(0.05785)
1109      ChWdParTableCol4[m].Draw()
1110
1111      ChWdParTableCanvas[m].Update()
1112
1113      GfxSave_PdfBookmark = "Title:%s" % (MeasNames[m])
1114      GfxSave_PdfFileName = "%s/AmpScanFitTable.pdf" % (PdfDir)
1115      if m ==     0:
1116          GfxSave_PdfFileName = "%s/AmpScanFitTable.pdf(" % (PdfDir)
1117      if m == _nM-1:
1118          GfxSave_PdfFileName = "%s/AmpScanFitTable.pdf)" % (PdfDir)
1119      if ExportPdf:
1120          ChWdParTableCanvas[m].Print(GfxSave_PdfFileName,GfxSave_PdfBookmark)
1121      GfxSave_EpsFileName = "%s/%s/_%s_AmpScanFitTable.eps"\
1122                            % (EpsDirectory, MeasNames[m], MeasNames[m])
1123      if ExportEps:
1124          ChWdParTableCanvas[m].SaveAs(GfxSave_EpsFileName)
1125
1126      PlotCount += 1
```

```
1127
1128 LogIt("INFO - %4s tables generated" % PlotCount)
1129
1130 ############################ Parameter versus wDAC
#############################
1131
1132 LogIt("---- GENERATING WIDTH SCAN HISTOGRAMS ---- ")
1133
1134 PlotCount = 0
1135
1136 ChAmpHisMinMax = []
1137 for m in range(0, _nM):
1138     ChAmpHisMinMax.append([])
1139     for a in range(0, _nA):
1140         ChAmpHisMinMax[m].append([])
1141         for p in range(0, 3):
1142             ChAmpHisMinMax[m][a].append([])
1143             ChAmpHisMinMax[m][a][p]=array('d')
1144             DeltaQ              = max(_ChAmpFit[m][a][p]) - min(_ChAmpFit[m][a][p])
1145             ChAmpHisMinMax[m][a][p].append(min(_ChAmpFit[m][a][p])\
1146                                            - DeltaQ*additionalrate)
1147             ChAmpHisMinMax[m][a][p].append(max(_ChAmpFit[m][a][p])\
1148                                            + DeltaQ*(additionalrate+diffrate))
1149
1150 hChAmp = []
1151 _ChAmpFitPar=[]
1152 for m in range(0, _nM):
1153     hChAmp.append([])
1154     _ChAmpFitPar.append([])
1155     for a in range(0, _nA):
1156         hChAmp[m].append([])
1157         _ChAmpFitPar[m].append([])
1158         for p in range(0, 3):
1159             _ChAmpFitPar[m][a].append([])
1160             hChAmp[m][a].append([])
1161             hChAmp[m][a][p] = TH1F("h%i_A0x%02X_p%s"\
1162                       % (m, aDACMultiGfx[a], ParaName[p]),"",\
1163                       q_nbin,ChAmpHisMinMax[m][a][p][0],ChAmpHisMinMax[m][a][p][1])
1164             for value in _ChAmpFit[m][a][p]:
1165                 hChAmp[m][a][p].Fill(value)
1166             hChAmp[m][a][p].Fit("gaus","Q")
1167
1168             fitLine = hChAmp[m][a][p].FindObject("gaus")
1169             fitLine.SetName("m%ia0x%02X%s" % (m,aDACMultiGfx[a],ParaName[p]))
1170             _Const = fitLine.GetParameter(0)
1171             _Mean  = fitLine.GetParameter(1)
1172             _Sigma = fitLine.GetParameter(2)
1173
1174             _ChAmpFitPar[m][a][p].append(_Const)
1175             _ChAmpFitPar[m][a][p].append(_Mean)
1176             _ChAmpFitPar[m][a][p].append(_Sigma)
1177
1178 for m in range(0, _nM):
1179     for a in range(0, _nA):
1180         for p in range(0, 3):
1181             GfxTitle = "%s %s @ aDAC=0x%02X"\
1182                       % (hBaseTitle[1][m], ParaName[p], aDACMultiGfx[a])
1183             x = hChAmp[m][a][p].GetXaxis()
1184             x.SetTitleOffset(v1_)
1185             x.SetLabelFont(v3_)
1186             x.SetLabelSize(v4_)
1187             y = hChAmp[m][a][p].GetYaxis()
1188             y.SetTitleOffset(v1_)
1189             y.SetLabelFont(v3_)
1190             y.SetLabelSize(v4_)
1191             hChAmp[m][a][p].SetLineColor(v2_)
```

```
1192             hChAmp[m][a][p].SetLineWidth(v5_)
1193             hChAmp[m][a][p].SetFillColor(v6_)
1194             hChAmp[m][a][p].SetMarkerSize(v7_)
1195             hChAmp[m][a][p].SetMarkerStyle(v8_)
1196             hChAmp[m][a][p].SetTitle(GfxTitle)
1197             hChAmp[m][a][p].SetXTitle(HisLegX[1][m][p])
1198             hChAmp[m][a][p].SetYTitle(HisLegY)
1199
1200 _ChAmpFitCanvas=[]
1201 _mA = min(HistLimit, _nA)
1202 for m in range(0, _nM):
1203     _ChAmpFitCanvas.append([])
1204     for a in range(0, _mA):
1205         _ChAmpFitCanvas[m].append([])
1206         n = a
1207         if HisSingleCanvas: n = 0
1208         _ChAmpFitCanvas[m][n] = TCanvas("m%i_a0x%02X"\
1209                                         % (m,aDACMultiGfx[a]),name3_can-
vas,1395,450)
1210         _ChAmpFitCanvas[m][n].Divide(3,1)
1211         for p in range(0, 3):
1212             _ChAmpFitCanvas[m][n].cd(p+1)
1213             hChAmp[m][a][p].Draw()
1214
1215         _ChAmpFitCanvas[m][n].Update()
1216
1217         GfxSave_PdfBookmark = "Title:%s_Amp0x%02X"%(MeasNames[m], aDACMul-
tiGfx[a])
1218         GfxSave_PdfFileName = "%s/%sWidScanFitHist.pdf" % (PdfDir, MeasNames[m])
1219         if a ==     0:
1220             GfxSave_PdfFileName = "%s/%sWidScanFitHist.pdf("%(PdfDir, MeasNames[m])
1221         if a == _mA-1:
1222             GfxSave_PdfFileName = "%s/%sWidScanFitHist.pdf)"%(PdfDir, MeasNames[m])
1223         if ExportPdf:
1224             _ChAmpFitCanvas[m][n].Print(GfxSave_PdfFileName,GfxSave_PdfBookmark)
1225         GfxSave_EpsFileName = "%s/%s/_%s_WidScanFitHist_A0x%02X.eps"\
1226                               % (EpsDirectory, MeasNames[m], MeasNames[m], aDACMul-
tiGfx[a])
1227         if ExportEps: _ChAmpFitCanvas[m][n].SaveAs(GfxSave_EpsFileName)
1228
1229         PlotCount += 1
1230
1231 LogIt("INFO - %4s histograms generated" % PlotCount)
1232
1233 ### Generate Parameters Table:
1234
1235 LogIt("------- DRAWING WIDTH SCAN TABLES -------- ")
1236
1237 PlotCount = 0
1238
1239 ChAmpParTableLabel = ["Area = p0 + p1*[wDAC]",\
1240                       "Charge = p0 + p1*[wDAC]",\
1241                       "Timing = p0 + p1*[wDAC]",\
1242                       "Width = p0 + p1*[wDAC]",\
1243                       "Peak = p0 + p1*[wDAC]"]
1244
1245 TableUnit = [["[binary]","[Volt*second]","[V*s / wDAC]","X^2/ndf"],\
1246              ["[binary]","[Coulomb]","[Coulomb / wDAC]","X^2/ndf"],\
1247              ["[binary]","[second]","[s / wDAC]","X^2/ndf"],\
1248              ["[binary]","[second]","[s / wDAC]","X^2/ndf"],\
1249              ["[binary]","[Volt]","[V / wDAC]","X^2/ndf"]]
```

```
1250
1251 _ChAmpErrorColor=[]
1252 for m in range(0, _nM):
1253     _ChAmpErrorColor.append([])
1254     for a in range(0, _nA):
1255         _ChAmpErrorColor[m].append([])
1256         _ChAmpErrorColor[m][a] = 1
1257         if _ChAmpWorstErr[m][a] > 1: _ChAmpErrorColor[m][a] = 874
1258         if _ChAmpWorstErr[m][a] > 2: _ChAmpErrorColor[m][a] = 2
1259
1260 ChAmpParTableCanvas=[]
1261 ChAmpParTableTitle=[]
1262 ChAmpParTableCol1=[]
1263 ChAmpParTableCol2=[]
1264 ChAmpParTableCol3=[]
1265 ChAmpParTableCol4=[]
1266
1267 for m in range(0, _nM):
1268     ChAmpParTableCanvas.append([])
1269     ChAmpParTableTitle.append([])
1270     ChAmpParTableCol1.append([])
1271     ChAmpParTableCol2.append([])
1272     ChAmpParTableCol3.append([])
1273     ChAmpParTableCol4.append([])
1274     ChAmpParTableCanvas[m] = TCanvas("WidScanParTable_m%s"\
1275                     % (m), "%s Table for f(wDAC)" % MeasNames[m], 700, 630)
1276
1277     ChAmpParTableTitle[m] = TPaveText(.25,.8867,.75,.9699)
1278     ChAmpParTableTitle[m].AddText(ChAmpParTableLabel[m])
1279     #ChAmpParTableTitle.SetTextSize(0.05)
1280     ChAmpParTableTitle[m].Draw()
1281
1282     ChAmpParTableCol1[m] = TPaveText(.02,.03,.155,.864)
1283     ChAmpParTableCol1[m].AddText("aDAC")
1284     ChAmpParTableCol1[m].AddLine(.0,(_nA+1)*1./(_nA+2),1.,(_nA+1)*1./(_nA+2))
1285     ChAmpParTableCol1[m].AddText(TableUnit[m][0])
1286     ChAmpParTableCol1[m].AddLine(.0,(_nA-0)*1./(_nA+2),1.,(_nA-0)*1./(_nA+2))
1287     for a in range(0, _nA):
1288         lastrow = ChAmpParTableCol1[m].AddText("0x%02X" % (aDACMultiGfx[a]))
1289         lastrow.SetTextColor(_ChAmpErrorColor[m][a])
1290     ChAmpParTableCol1[m].SetTextSize(0.03215)
1291     ChAmpParTableCol1[m].Draw()
1292
1293     ChAmpParTableCol2[m] = TPaveText(.155,.03,.489,.864)
1294     ChAmpParTableCol2[m].AddText("p0 Mean #pm Sigma")
1295     ChAmpParTableCol2[m].AddLine(.0,(_nA+1)*1./(_nA+2),1.,(_nA+1)*1./(_nA+2))
1296     ChAmpParTableCol2[m].AddText(TableUnit[m][1])
1297     ChAmpParTableCol2[m].AddLine(.0,(_nA-0)*1./(_nA+2),1.,(_nA-0)*1./(_nA+2))
1298     for a in range(0, _nA):
1299         lastrow = ChAmpParTableCol2[m].AddText("%.3E #pm %.3E"\
1300                         %(_ChAmpFitPar[m][a][0][1], _ChAmpFitPar[m][a][0][2]))
1301         lastrow.SetTextColor(_ChAmpErrorColor[m][a])
1302     ChAmpParTableCol2[m].SetTextSize(0.03215)
1303     ChAmpParTableCol2[m].Draw()
1304
1305     ChAmpParTableCol3[m] = TPaveText(.489,.03,.823,.864)
1306     ChAmpParTableCol3[m].AddText("p1 Mean #pm Sigma")
1307     ChAmpParTableCol3[m].AddLine(.0,(_nA+1)*1./(_nA+2),1.,(_nA+1)*1./(_nA+2))
1308     ChAmpParTableCol3[m].AddText(TableUnit[m][2])
1309     ChAmpParTableCol3[m].AddLine(.0,(_nA-0)*1./(_nA+2),1.,(_nA-0)*1./(_nA+2))
1310     for a in range(0, _nA):
1311         lastrow = ChAmpParTableCol3[m].AddText("%.3E #pm %.3E"\
1312                         %(_ChAmpFitPar[m][a][1][1], _ChAmpFitPar[m][a][1][2]))
1313         lastrow.SetTextColor(_ChAmpErrorColor[m][a])
1314     ChAmpParTableCol3[m].SetTextSize(0.03215)
1315     ChAmpParTableCol3[m].Draw()
1316
1317     ChAmpParTableCol4[m] = TPaveText(.823,.03,.98,.864)
```

```
1318     ChAmpParTableCol4[m].AddText("Max Err")
1319     ChAmpParTableCol4[m].AddLine(.0,(_nA+1)*1./(_nA+2),1.,(_nA+1)*1./(_nA+2))
1320     ChAmpParTableCol4[m].AddText(TableUnit[m][3])
1321     ChAmpParTableCol4[m].AddLine(.0,(_nA-0)*1./(_nA+2),1.,(_nA-0)*1./(_nA+2))
1322     _ErrorColor = 1
1323     for a in range(0, _nA):
1324         lastrow = ChAmpParTableCol4[m].AddText("%.3E"%(_ChAmpWorstErr[m][a]))
1325         lastrow.SetTextColor(_ChAmpErrorColor[m][a])
1326     ChAmpParTableCol4[m].SetTextSize(0.03215)
1327     ChAmpParTableCol4[m].Draw()
1328 
1329     ChAmpParTableCanvas[m].Update()
1330 
1331     GfxSave_PdfBookmark = "Title:%s" % (MeasNames[m])
1332     GfxSave_PdfFileName = "%s/WidScanFitTable.pdf" % (PdfDir)
1333     if m ==     0:
1334         GfxSave_PdfFileName = "%s/WidScanFitTable.pdf(" % (PdfDir)
1335     if m == _nM-1:
1336         GfxSave_PdfFileName = "%s/WidScanFitTable.pdf)" % (PdfDir)
1337     if ExportPdf:
1338         ChAmpParTableCanvas[m].Print(GfxSave_PdfFileName,GfxSave_PdfBookmark)
1339     GfxSave_EpsFileName = "%s/%s/_%s_WidScanFitTable.eps"\
1340                           % (EpsDirectory, MeasNames[m], MeasNames[m])
1341     if ExportEps:
1342         ChAmpParTableCanvas[m].SaveAs(GfxSave_EpsFileName)
1343 
1344     PlotCount += 1
1345 
1346 LogIt("INFO - %4s tables generated" % PlotCount)
1347 
1348 ###############################################################################
1349 ##                                                                           ##
1350 ##          Generating Histograms of Timing Distribution Along Channels      ##
1351 ##                                                                           ##
1352 ###############################################################################
1353 
1354 LogIt("------ GENERATING TIMING HISTOGRAMS ------ ")
1355 
1356 ### PLOTING Timing Histograms:
1357 
1358 v1_ = 1.5 #title offset
1359 v2_ = 4   #line color
1360 v3_ = 43  #label font
1361 v4_ = 15  #label size
1362 v5_ = 3   #line width
1363 v6_ = 16  #fill color
1364 v7_ = 1   #marker size
1365 v8_ = 20  #marker style
1366 
1367 ## Filter by Amplitude
1368 
1369 name1_canvas = "Evaluation of the Temporal Alignment"
1370 
1371 legendax = "mean 3mV Trigger Time [ns]"
1372 legenday = "number of [Channel*aDAC] per bin"
1373 
1374 nWidthsTiming = 6
1375 sWidthsTiming = 3
1376 
1377 HistogramCut = 255 - (NumberOfAmplitudes*17) + 1
1378 
1379 hWdT = []
1380 hWdMinAmp = []
1381 n  = 0
1382 for i in range(0, len(_Data[2])):
1383     # Fill proper histogram:
1384     if _Data[2][i] > HistogramCut:
1385         for j in range(0, nWidthsTiming):
1386             if n == 0:
```

```
1387                 hWdT.append([])
1388                 hWdMinAmp.append([])
1389                 hWdT[j] = TH1F("hT_W0x%02X"\
1390                                % (sWidthsTiming+j),"",nbin,mintiming,maxtiming)
1391                 hWdMinAmp[j]=255
1392             if _Data[1][i] == sWidthsTiming+j:
1393                 hWdT[j].Fill(_Data[12][i]*1e+9)
1394                 if _Data[2][i] <= hWdMinAmp[j]: hWdMinAmp[j] = _Data[2][i]
1395         n =+ 1
1396
1397 hWdTtitle = []
1398 for j in range(0, nWidthsTiming):
1399     hWdTtitle.append("Timing @ wDAC 0x%02X & Amp #geq %i"\
1400                      % (sWidthsTiming+j, hWdMinAmp[j]))
1401
1402 for j in range(0, nWidthsTiming):
1403     x = hWdT[j].GetXaxis()
1404     x.SetTitleOffset(v1_)
1405     x.SetLabelFont(v3_)
1406     x.SetLabelSize(v4_)
1407     y = hWdT[j].GetYaxis()
1408     y.SetTitleOffset(v1_)
1409     y.SetLabelFont(v3_)
1410     y.SetLabelSize(v4_)
1411     hWdT[j].SetLineColor(v2_)
1412     hWdT[j].SetLineWidth(v5_)
1413     hWdT[j].SetFillColor(v6_)
1414     hWdT[j].SetMarkerSize(v7_)
1415     hWdT[j].SetMarkerStyle(v8_)
1416     hWdT[j].SetTitle(hWdTtitle[j])
1417     hWdT[j].SetXTitle(legendax)
1418     hWdT[j].SetYTitle(legenday)
1419
1420 canv1 = TCanvas("canv1",name1_canvas,1400,900)
1421 canv1.Divide(3,2)
1422 for j in range(0, nWidthsTiming):
1423     canv1.cd(j+1)
1424     hWdT[j].Fit("landau","Q")
1425     hWdT[j].Draw()
1426
1427 if ExportPdf: canv1.SaveAs("%s/TimingDistribution_wDAC.pdf" % (PdfDir))
1428 if ExportEps: canv1.SaveAs("%s/TimingDistribution_wDAC.eps" % (EpsDirectory))
1429
1430 ## Filter by Width
1431
1432 name2_canvas = "Evaluation of the Temporal Alignment"
1433 legendax = "mean 3mV Trigger Time [ns]"
1434 legenday = "number of [Channel*wDAC] per bin"
1435 hAmpMin = 1.e15
1436 hAmpMax = -1.e15
1437 n_aDACChCut = 10
1438
1439 for a in range(n_aDACChCut, _nA):
1440     for c in range(0, _nC):
1441         for tValue in rdChAmp[a][c][12]:
1442             if tValue < hAmpMin: hAmpMin = tValue
1443             if tValue > hAmpMax: hAmpMax = tValue
1444
1445 _delta = hAmpMax - hAmpMin
1446 hAmpHistMin = hAmpMin - _delta*additionalrate
1447 hAmpHistMax = hAmpMax + _delta*(additionalrate+diffrate)
1448
1449 hAmpT = []
1450 n  = 0
1451 for a in range(n_aDACChCut, _nA):
```

```
1452     # Fill proper histogram:
1453     for c in range(0, _nC):
1454         m  = 0
1455         for tValue in rdChAmp[a][c][12]:
1456             if m == 0 and c == 0:
1457                 hAmpT.append([])
1458                 hAmpT[n] = TH1F("hT_A0x%02X"\
1459                     % (aDACMultiGfx[a]),"",nbin,hAmpHistMin*1e+9,hAmpHist-
Max*1e+9)
1460             hAmpT[n].Fill(tValue*1e+9)
1461             m =+ 1
1462     n += 1
1463
1464 hAmpTtitle = []
1465 for j in range(0, n):
1466     hAmpTtitle.append("Timing @ aDAC 0x%02X #forall wDAC"\
1467                     % (aDACMultiGfx[j+n_aDAChCut]))
1468
1469 for j in range(0, n):
1470     x = hAmpT[j].GetXaxis()
1471     x.SetTitleOffset(v1_)
1472     x.SetLabelFont(v3_)
1473     x.SetLabelSize(v4_)
1474     y = hAmpT[j].GetYaxis()
1475     y.SetTitleOffset(v1_)
1476     y.SetLabelFont(v3_)
1477     y.SetLabelSize(v4_)
1478     hAmpT[j].SetLineColor(v2_)
1479     hAmpT[j].SetLineWidth(v5_)
1480     hAmpT[j].SetFillColor(v6_)
1481     hAmpT[j].SetMarkerSize(v7_)
1482     hAmpT[j].SetMarkerStyle(v8_)
1483     hAmpT[j].SetTitle(hAmpTtitle[j])
1484     hAmpT[j].SetXTitle(legendax)
1485     hAmpT[j].SetYTitle(legenday)
1486
1487 canv2 = TCanvas("canv2",name2_canvas,1400,900)
1488 canv2.Divide(3,2)
1489 for j in range(0, n):
1490     canv2.cd(j+1)
1491     hAmpT[j].Fit("landau","Q")
1492     hAmpT[j].Draw()
1493
1494 if ExportPdf: canv2.SaveAs("%s/TimingDistribution_aDAC.pdf" % (PdfDir))
1495 if ExportEps: canv2.SaveAs("%s/TimingDistribution_aDAC.eps" % (EpsDirectory))
1496
1497 LogIt("------------      FINISH      ------------ ")
1498
1499 with open("%s/%s" % (LogDirectory, Log_file), 'w') as outputfile:
1500     for message in LogMessages:
1501         outputfile.write("%s\n" % (message))
1502
1503 with open("%s/%s" % (LogDirectory, Script_file), 'w') as outputfile:
1504     ThisScript = open(sys.argv[0])
1505     for line in ThisScript:
1506         outputfile.write(line)
1507
1508 with open("%s/%s" % (LogDirectory, ChWdFit_file), 'w') as outputfile:
1509     for m in range(0, len(_ChWdFit)):
1510         for w in range(0, len(_ChWdFit[m])):
1511             for p in range(0, len(_ChWdFit[m][w])):
1512                 outputfile.write("Measurement:%s, Width:0x%02X, Parameter:%s\n"\
1513                                  % (MeasNames[m], wDACMultiGfx[w], ParaName[p]))
1514                 for c in range(0, len(_ChWdFit[m][w][p])):
1515                     outputfile.write("%s," % (_ChWdFit[m][w][p][c]))
1516                 outputfile.write("\n")
```

```
1517
1518 with open("%s/%s" % (LogDirectory, ChAmpFit_file), 'w') as outputfile:
1519     for m in range(0, len(_ChAmpFit)):
1520         for a in range(0, len(_ChAmpFit[m])):
1521             for p in range(0, len(_ChAmpFit[m][a])):
1522                 outputfile.write("Measurement:%s, Amplitude:0x%02X, Parame-
ter:%s\n"\
1523                                  % (MeasNames[m], aDACMultiGfx[a], ParaName[p]))
1524                 for c in range(0, len(_ChAmpFit[m][a][p])):
1525                     outputfile.write("%s," % (_ChAmpFit[m][a][p][c]))
1526                 outputfile.write("\n")
```

Apêndice K

Histogramas da Seção 4.2.4

Índice de Histogramas deste Apêndice:

Figura 1 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x02$ 318
Figura 2 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x03$ 318
Figura 3 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x04$ 318
Figura 4 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x05$ 318
Figura 5 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x06$ 319
Figura 6 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x07$ 319
Figura 7 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x08$ 319
Figura 8 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x02$..... 320
Figura 9 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x03$..... 320
Figura 10 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x04$..... 320
Figura 11 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x05$..... 320
Figura 12 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x06$..... 321
Figura 13 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x07$..... 321
Figura 14 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x08$..... 321
Figura 15 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x02$..... 322
Figura 16 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x03$..... 322
Figura 17 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x04$..... 322
Figura 18 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x05$..... 322
Figura 19 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x06$..... 323
Figura 20 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x07$..... 323
Figura 21 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x08$..... 323

Figura 22 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x02$ 324
Figura 23 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x03$ 324
Figura 24 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x04$ 324
Figura 25 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x05$ 324
Figura 26 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x06$ 325
Figura 27 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x07$ 325
Figura 28 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x08$ 325
Figura 29 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x02$ 326
Figura 30 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x03$ 326
Figura 31 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x04$ 326
Figura 32 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x05$ 326
Figura 33 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x06$ 327
Figura 34 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x07$ 327
Figura 35 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x08$ 327
Figura 36 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x00$ 328
Figura 37 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x11$ 328
Figura 38 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x22$ 328
Figura 39 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x33$ 328
Figura 40 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x44$ 329
Figura 41 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x55$ 329
Figura 42 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x66$ 329
Figura 43 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x77$ 329
Figura 44 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x88$ 330
Figura 45 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x99$ 330
Figura 46 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0xAA$ 330
Figura 47 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0xBB$ 330
Figura 48 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0xCC$ 331
Figura 49 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0xDD$ 331

Figura 50 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xEE 331
Figura 51 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xFF 331
Figura 52 – Parâmetros de área em função de $wDAC$ @ $aDAC$ = 0x00 332
Figura 53 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x11 332
Figura 54 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x22 332
Figura 55 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x33 332
Figura 56 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x44 333
Figura 57 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x55 333
Figura 58 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x66 333
Figura 59 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x77 333
Figura 60 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x88 334
Figura 61 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0x99 334
Figura 62 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xAA 334
Figura 63 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xBB 334
Figura 64 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xCC 335
Figura 65 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xDD 335
Figura 66 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xEE 335
Figura 67 – Parâmetros de carga em função de $wDAC$ @ $aDAC$ = 0xFF 335
Figura 68 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x00 336
Figura 69 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x11 336
Figura 70 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x22 336
Figura 71 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x33 336
Figura 72 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x44 337
Figura 73 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x55 337
Figura 74 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x66 337
Figura 75 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x77 337
Figura 76 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x88 338
Figura 77 – Parâmetros de largura em função de $wDAC$ @ $aDAC$ = 0x99 338

Figura 78 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xAA 338
Figura 79 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xBB 338
Figura 80 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xCC 339
Figura 81 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xDD 339
Figura 82 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xEE 339
Figura 83 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xFF 339
Figura 84 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x00 340
Figura 85 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x11 340
Figura 86 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x22 340
Figura 87 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x33 340
Figura 88 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x44 341
Figura 89 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x55 341
Figura 90 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x66 341
Figura 91 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x77 341
Figura 92 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x88 342
Figura 93 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x99 342
Figura 94 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xAA .. 342
Figura 95 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xBB... 342
Figura 96 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xCC... 343
Figura 97 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xDD .. 343
Figura 98 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xEE... 343
Figura 99 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xFF ... 343
Figura 100 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x00 344
Figura 101 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x11 344
Figura 102 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x22 344
Figura 103 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x33 344
Figura 104 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x44 345
Figura 105 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x55 345

Figura 106 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0x66..... 345
Figura 107 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0x77..... 345
Figura 108 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0x88..... 346
Figura 109 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0x99..... 346
Figura 110 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0xAA..... 346
Figura 111 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0xBB..... 346
Figura 112 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0xCC..... 347
Figura 113 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0xDD..... 347
Figura 114 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0xEE..... 347
Figura 115 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC$ = 0xFF..... 347

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de área corrigida em função de $aDAC$:

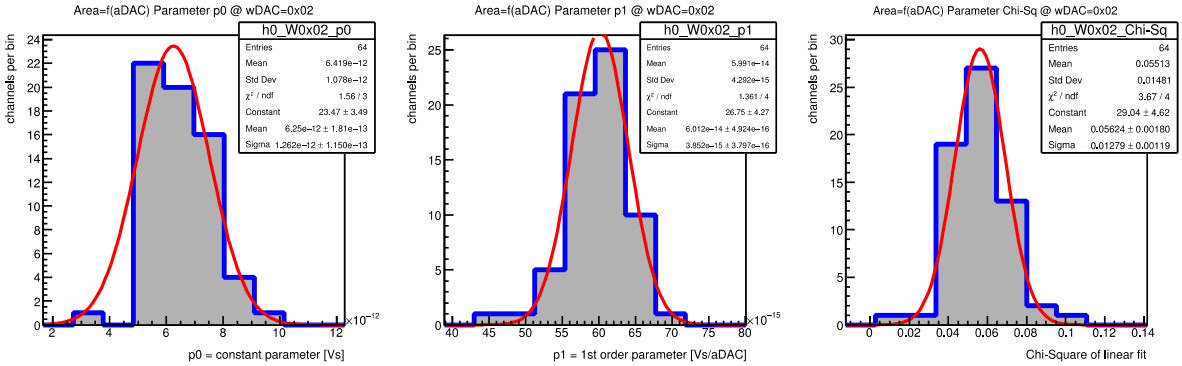


Figura 1 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x02$

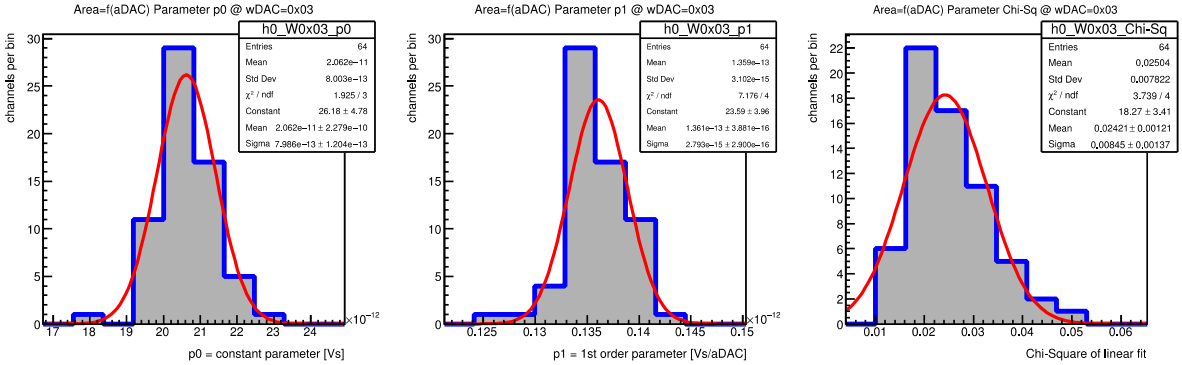


Figura 2 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x03$

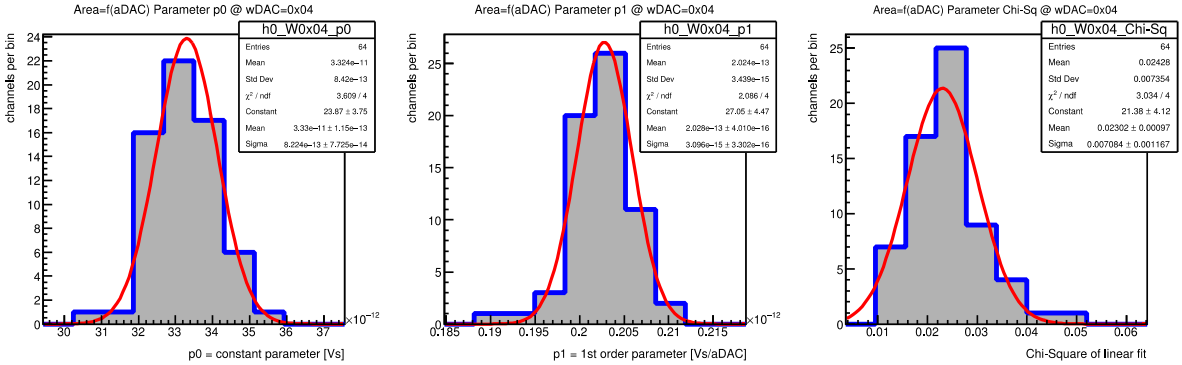


Figura 3 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x04$

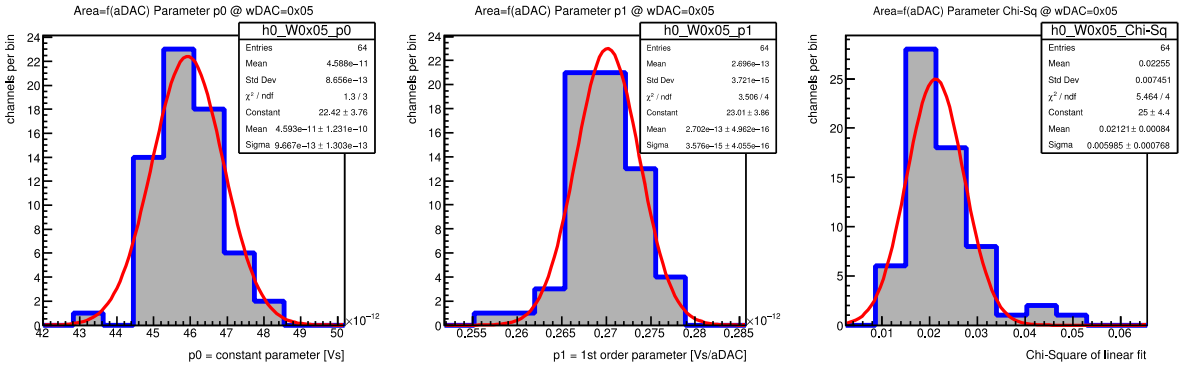


Figura 4 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0x05$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de área corrigida em função de $aDAC$ (continuação):

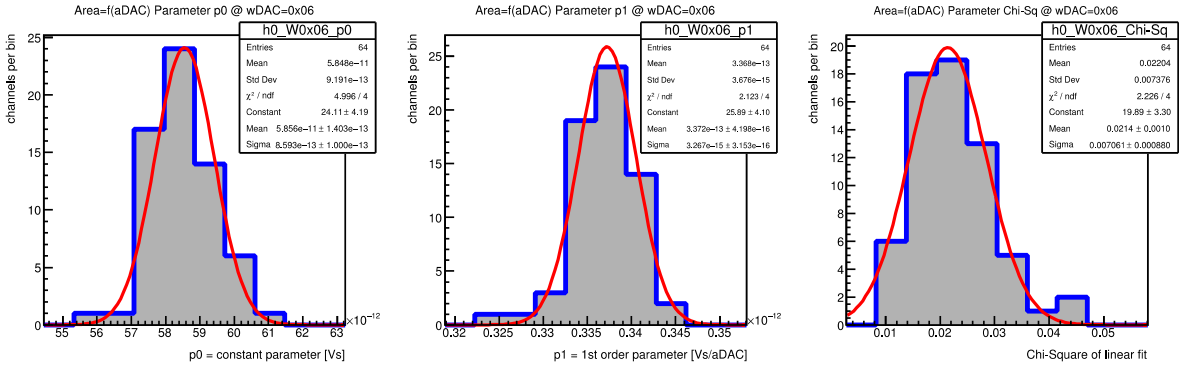


Figura 5 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0$x06

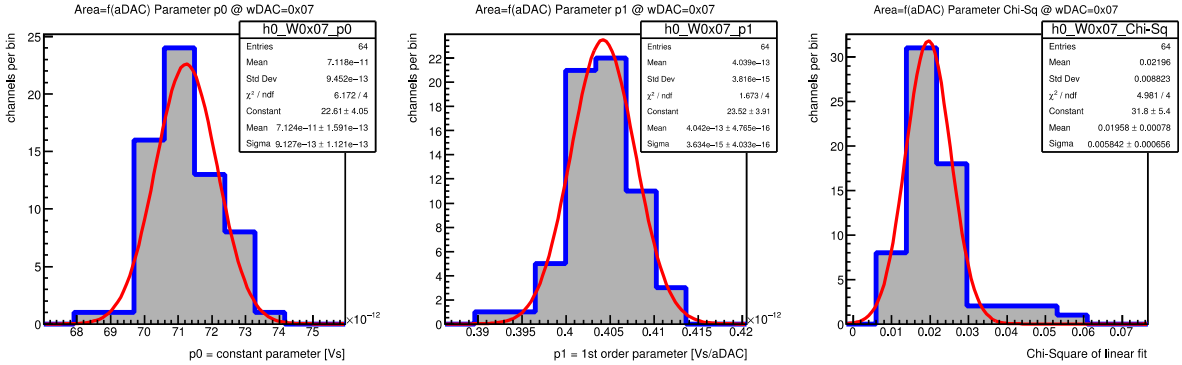


Figura 6 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0$x07

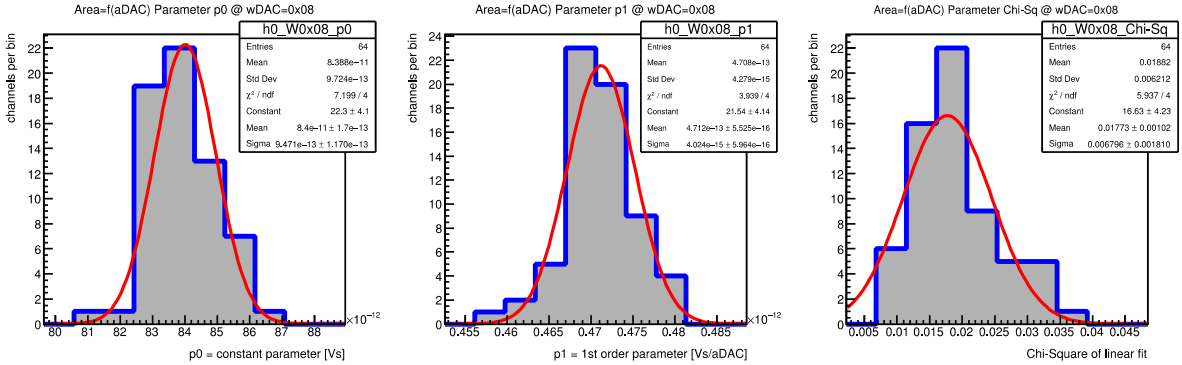


Figura 7 – Parâmetros da área em função de $aDAC$ @ $wDAC = 0$x08

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de carga calculada em função de $aDAC$:

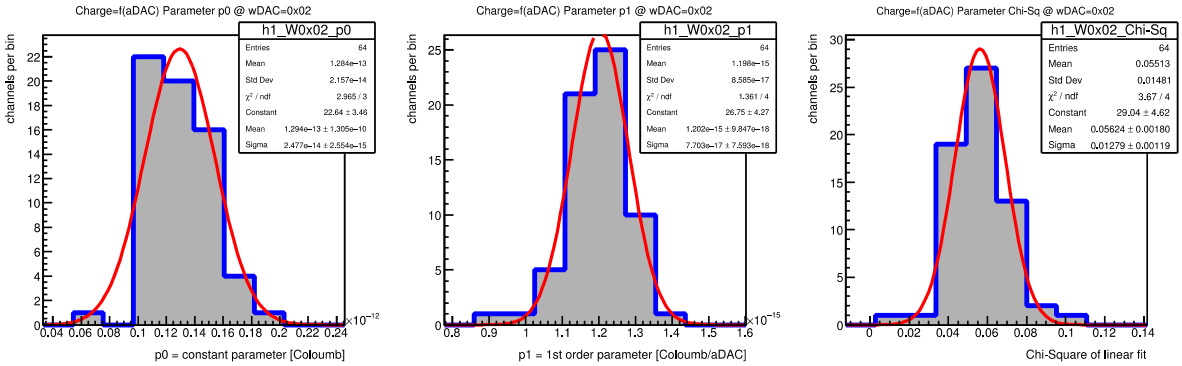


Figura 8 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x02$

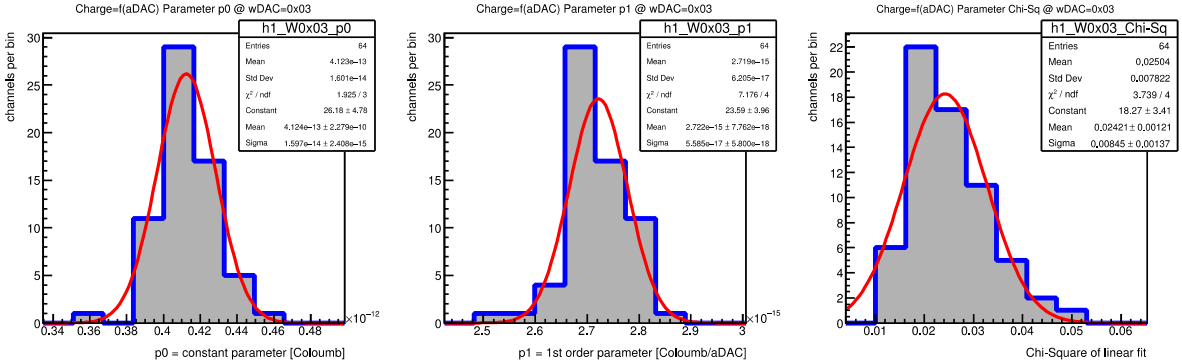


Figura 9 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x03$

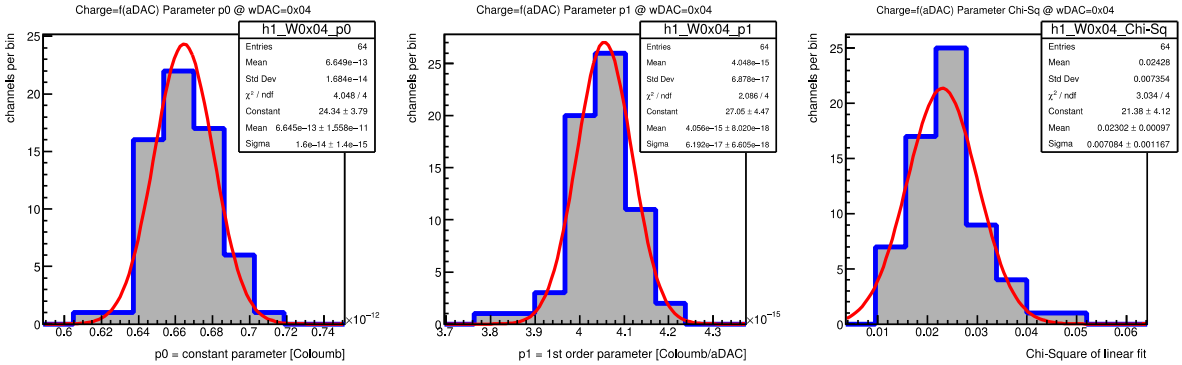


Figura 10 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x04$

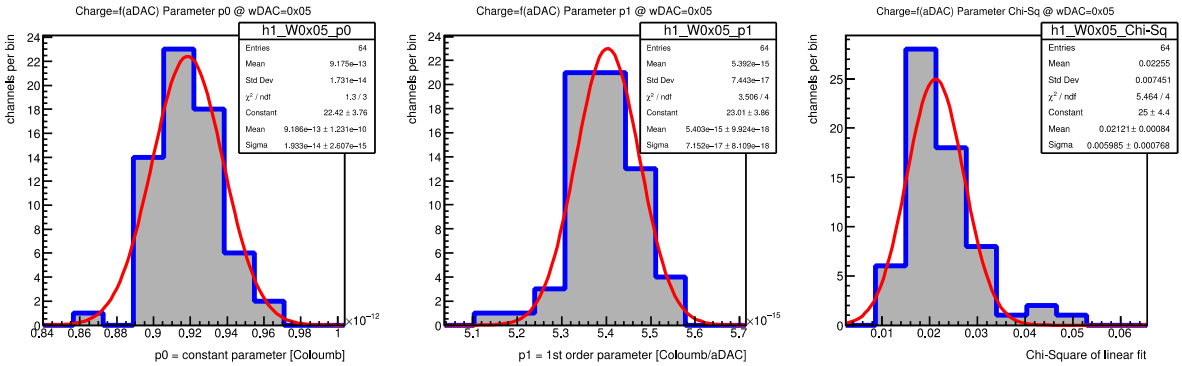


Figura 11 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x05$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de carga calculada em função de $aDAC$ (continuação):

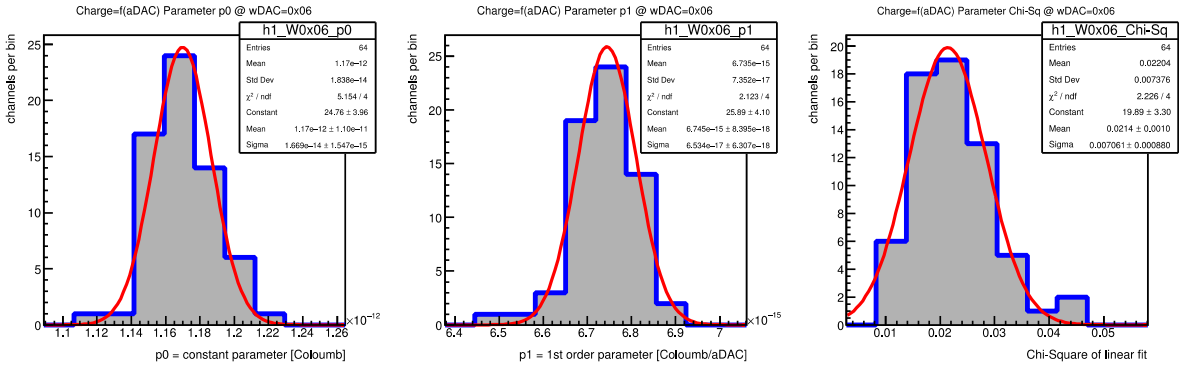


Figura 12 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x06$

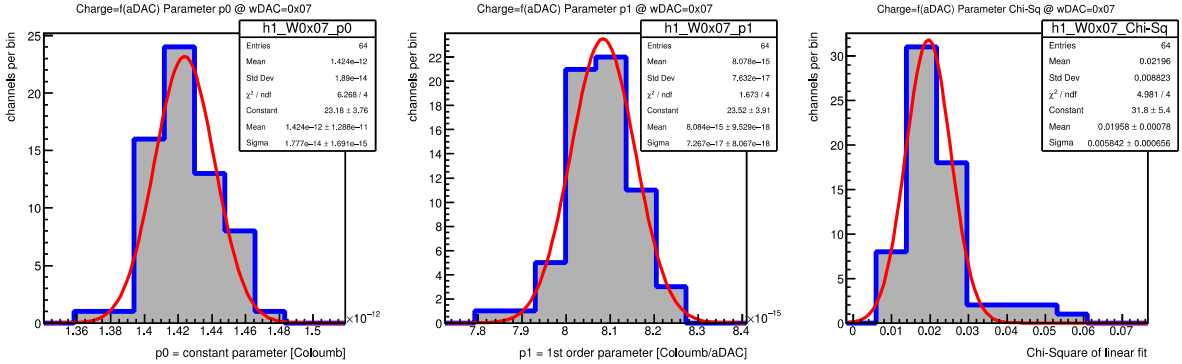


Figura 13 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x07$

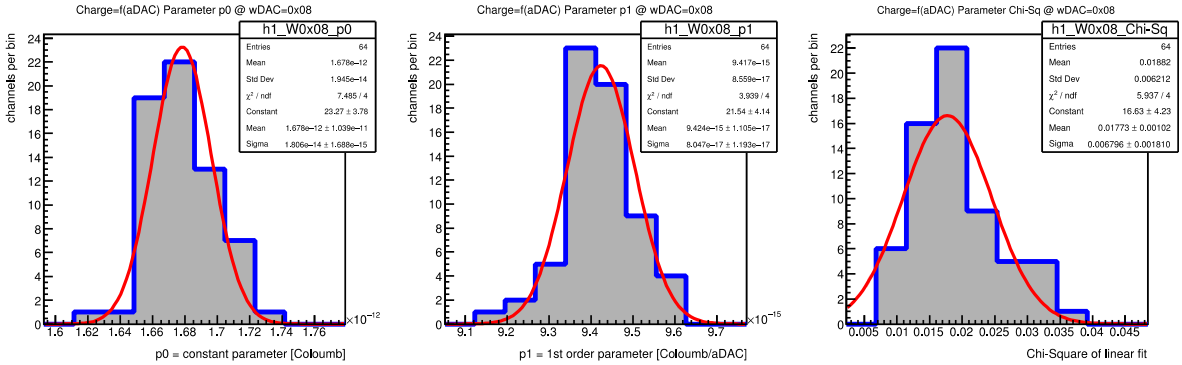


Figura 14 – Parâmetros da carga em função de $aDAC$ @ $wDAC = 0x08$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de largura do pulso em função de $aDAC$:

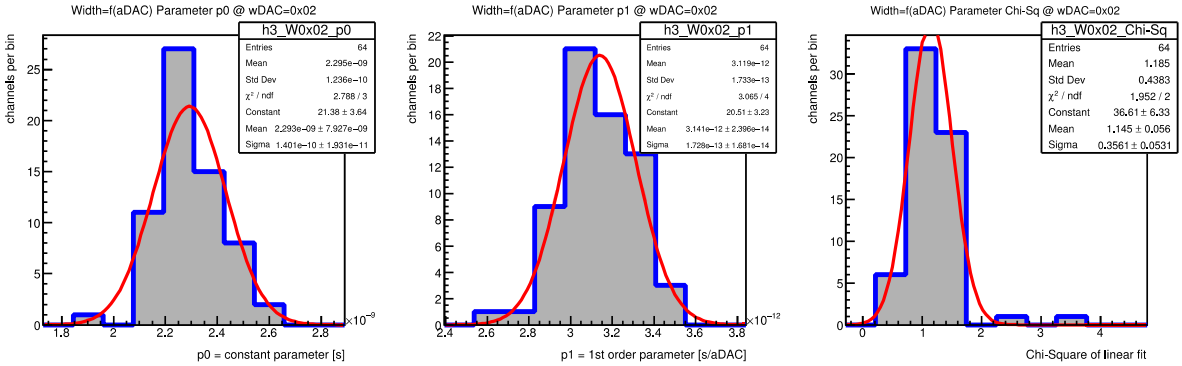


Figura 15 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x02$

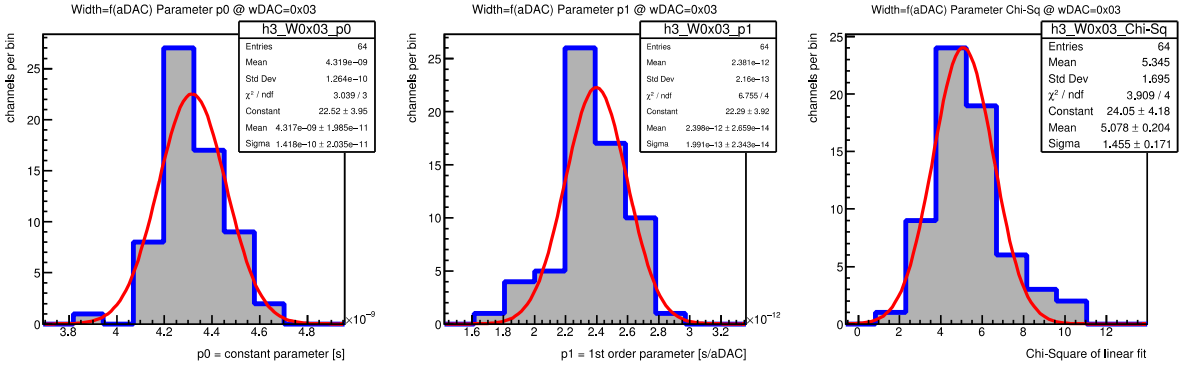


Figura 16 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x03$

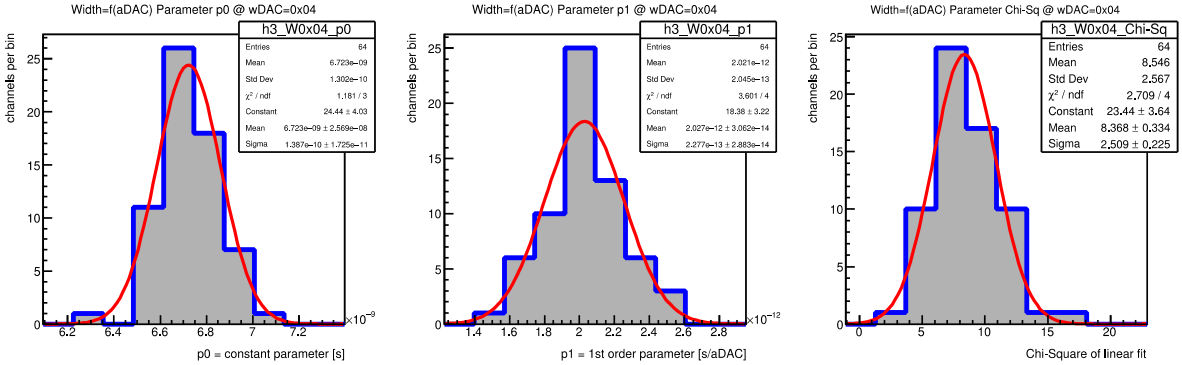


Figura 17 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x04$

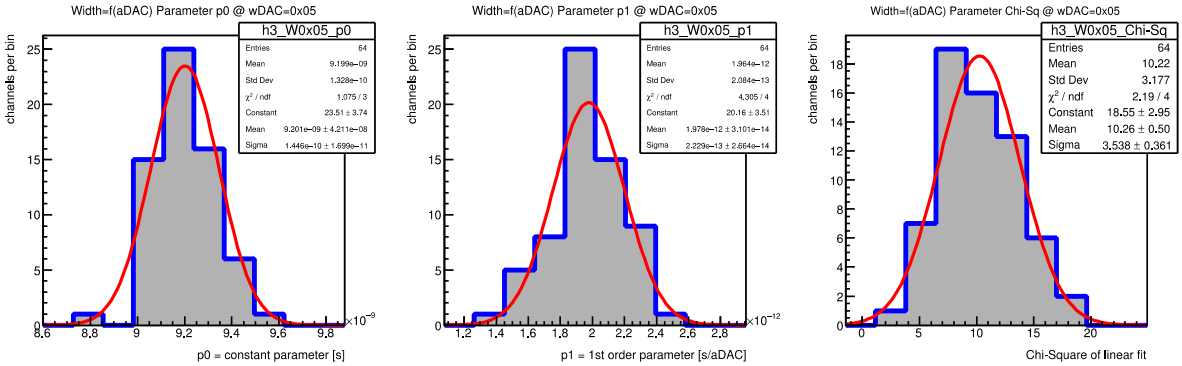


Figura 18 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x05$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de largura do pulso em função de $aDAC$ (continuação):

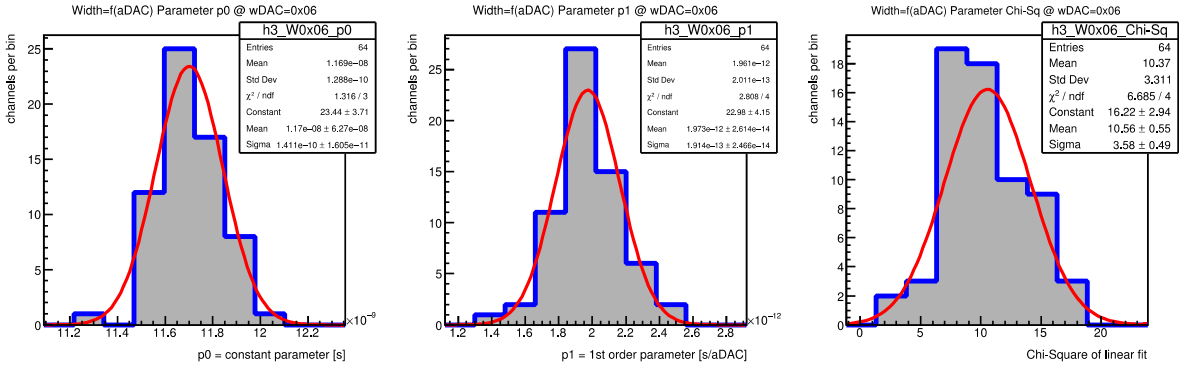


Figura 19 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x06$

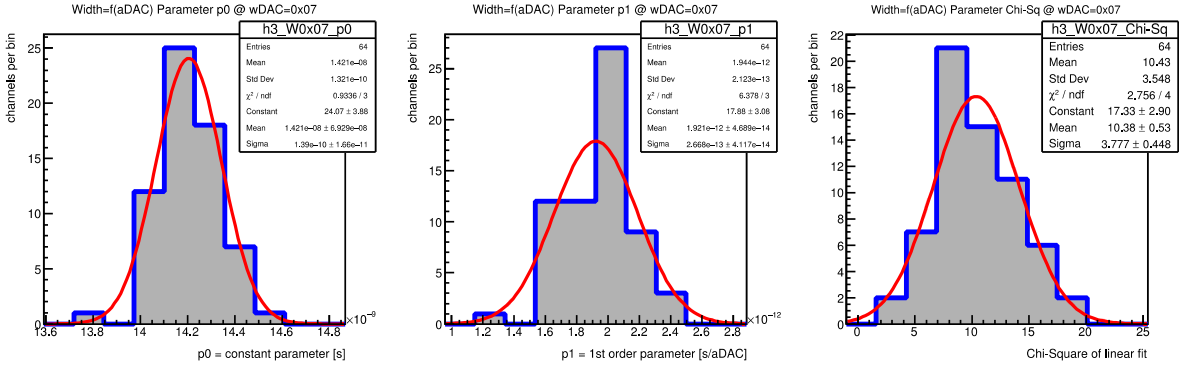


Figura 20 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x07$

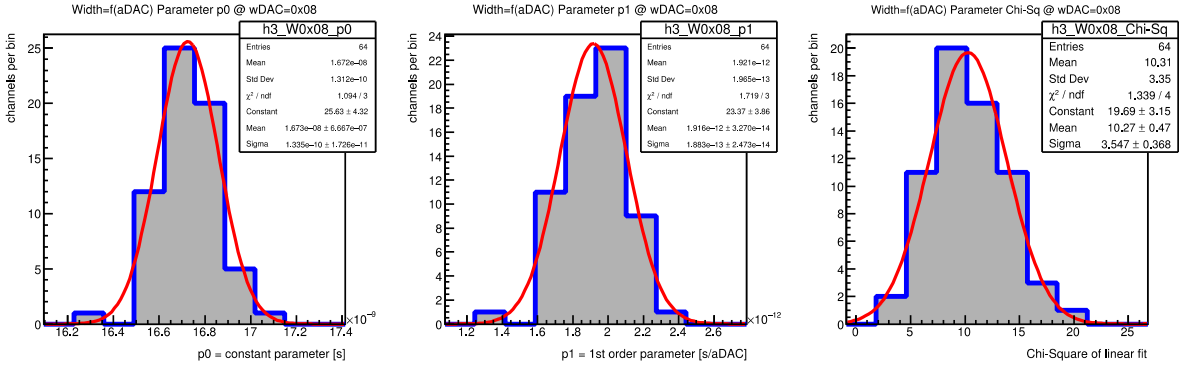


Figura 21 – Parâmetros da largura em função de $aDAC$ @ $wDAC = 0x08$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de temporização em função de $aDAC$:

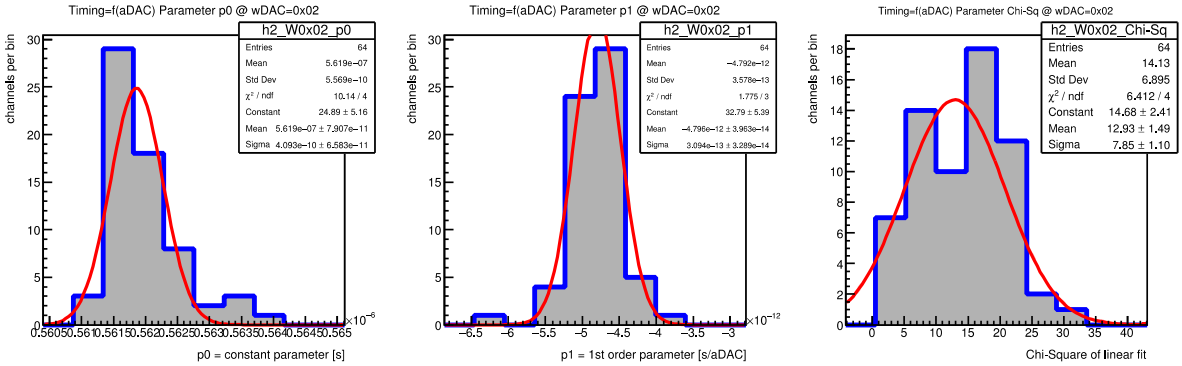


Figura 22 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x02$

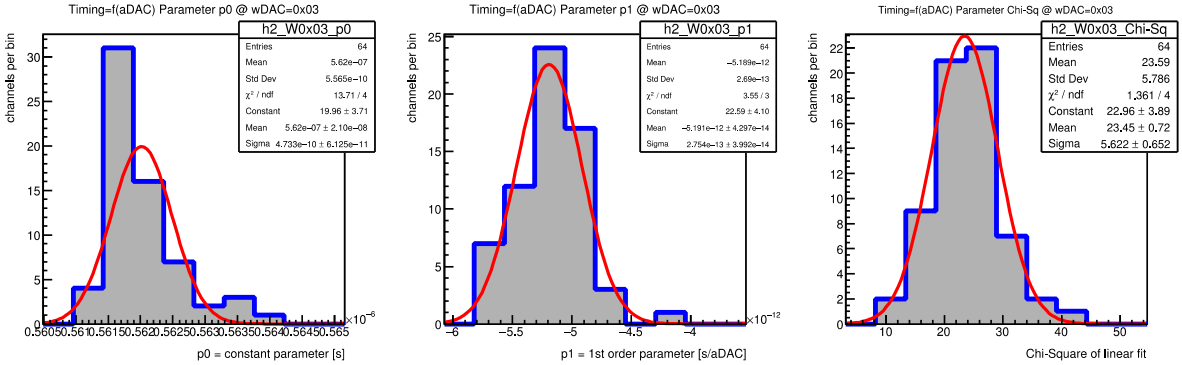


Figura 23 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x03$

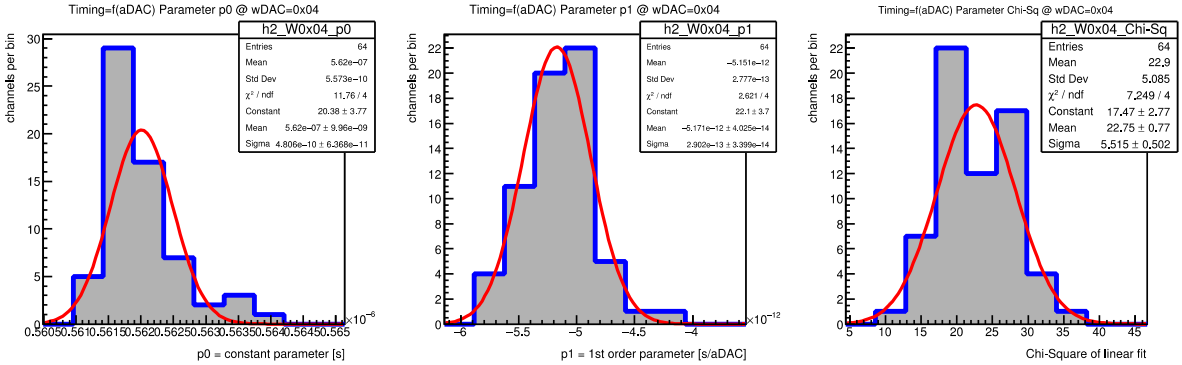


Figura 24 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x04$

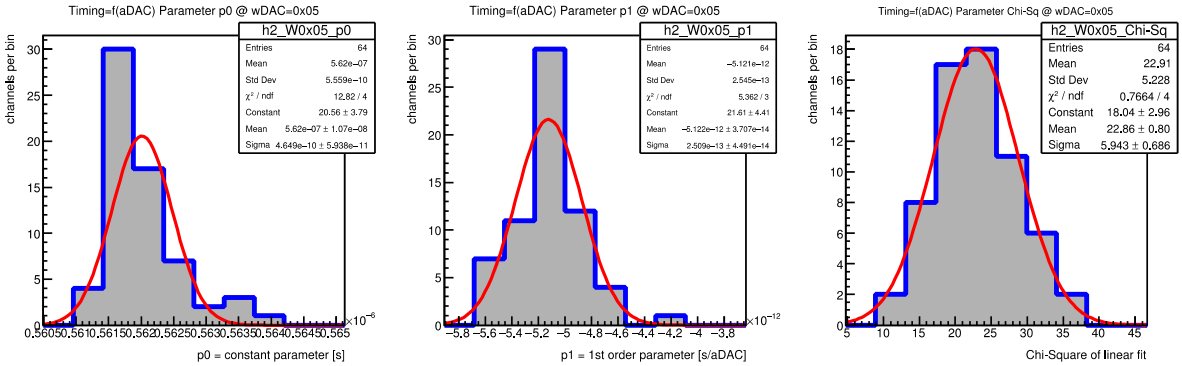


Figura 25 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x05$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de temporização em função de $aDAC$ (continuação):

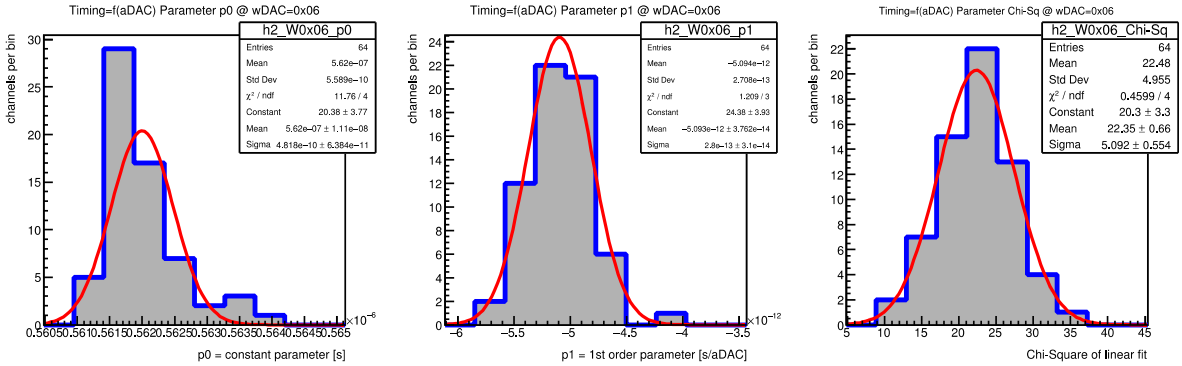


Figura 26 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x06$

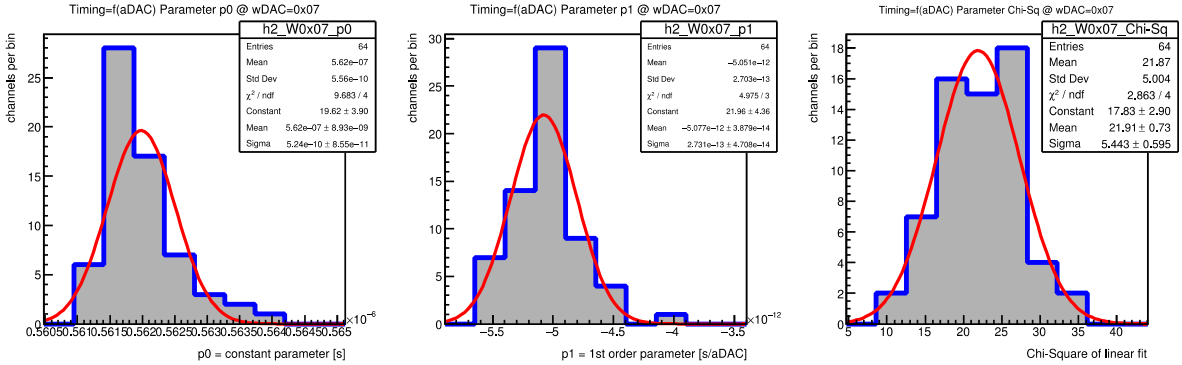


Figura 27 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x07$

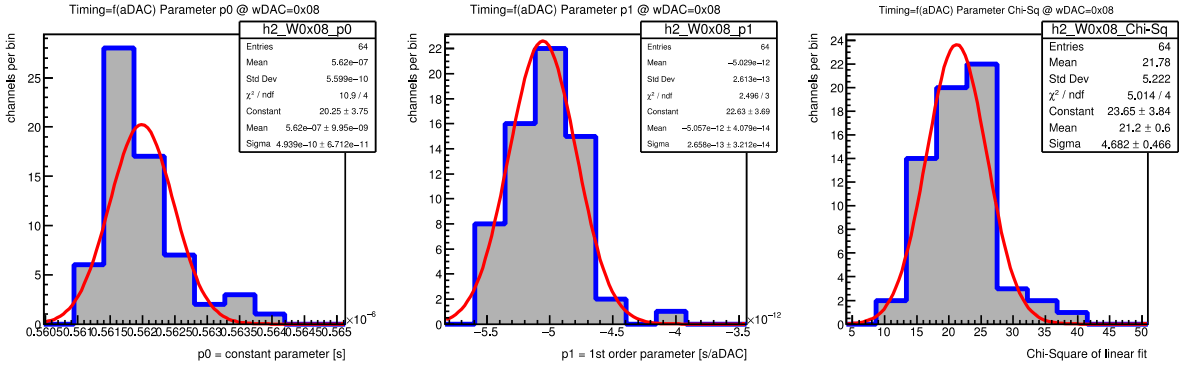


Figura 28 – Parâmetros da temporização em função de $aDAC$ @ $wDAC = 0x08$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de tensão de pico em função de $aDAC$:

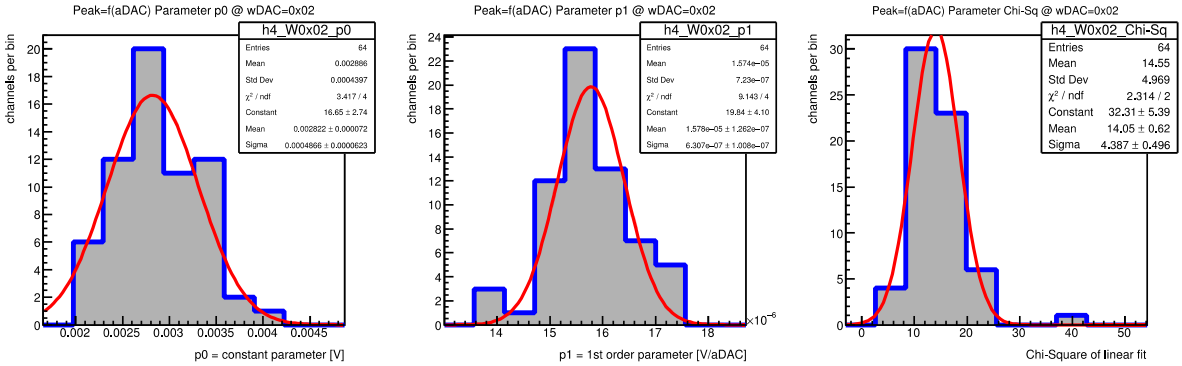


Figura 29 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x02$

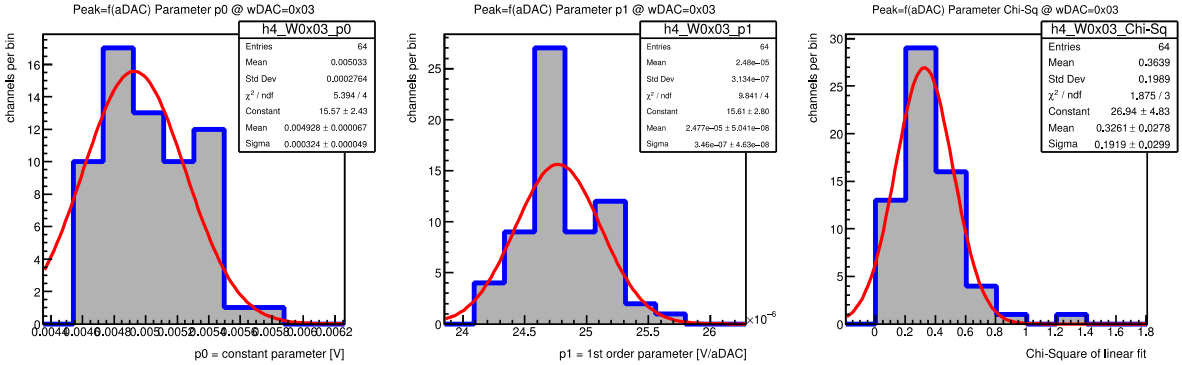


Figura 30 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x03$

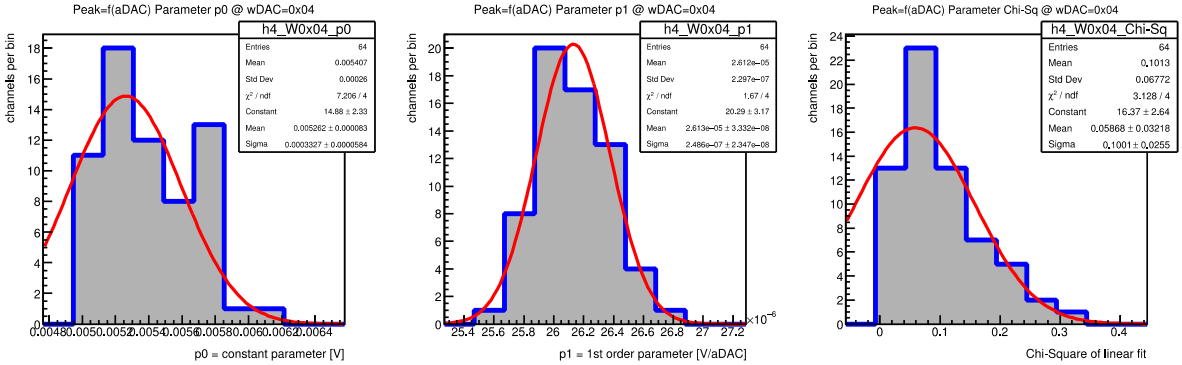


Figura 31 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x04$

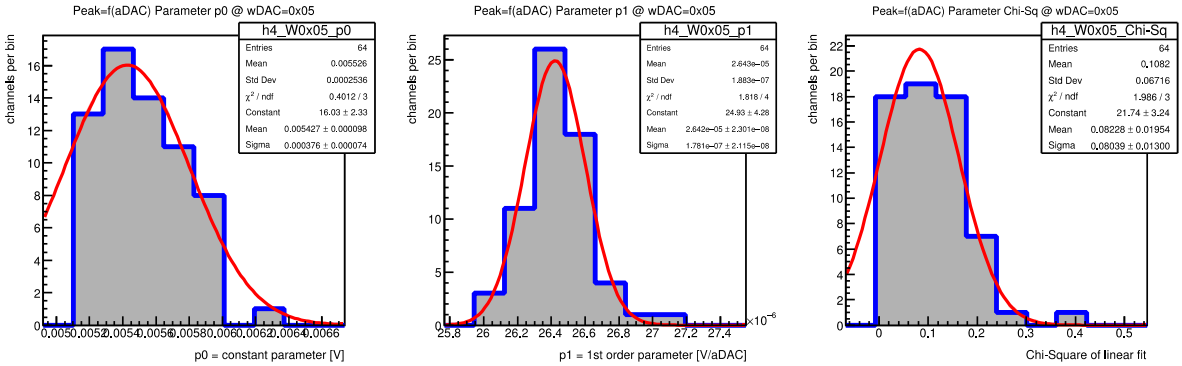


Figura 32 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x05$

- Distribuição em várias configurações fixas de $wDAC$ dos parâmetros de ajuste da curva de tensão de pico em função de $aDAC$ (continuação):

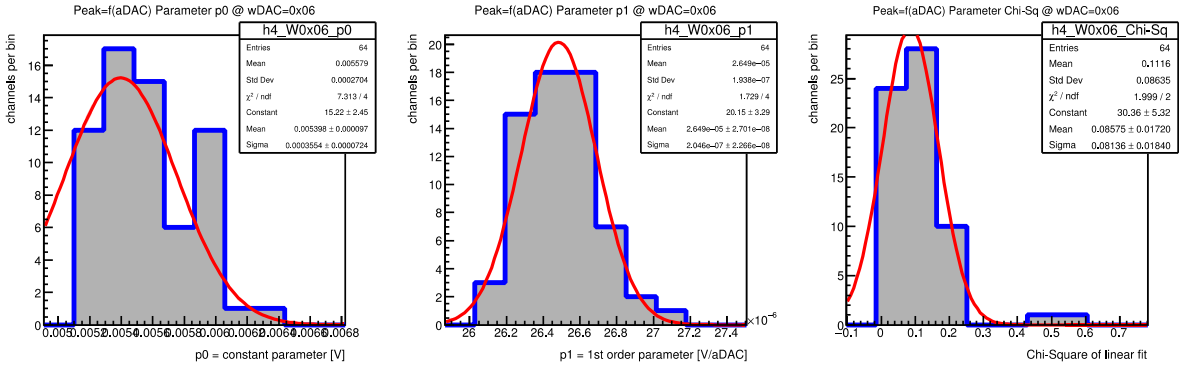


Figura 33 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x06$

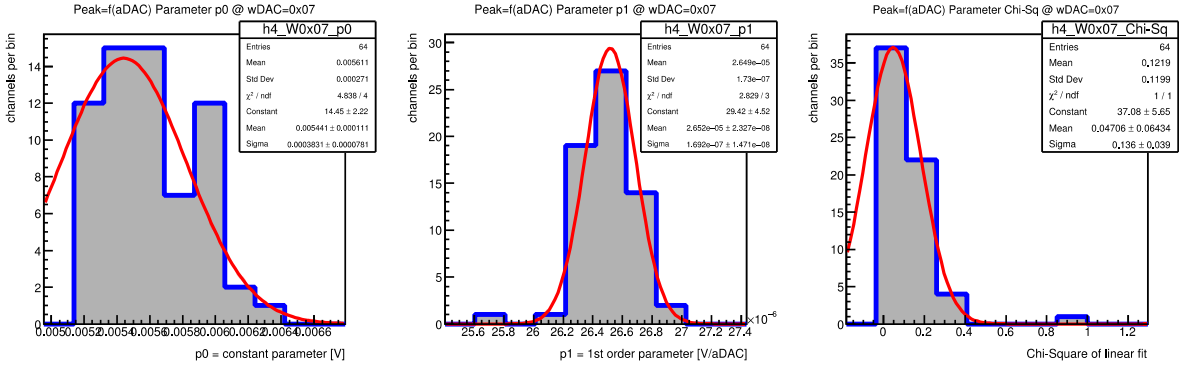


Figura 34 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x07$

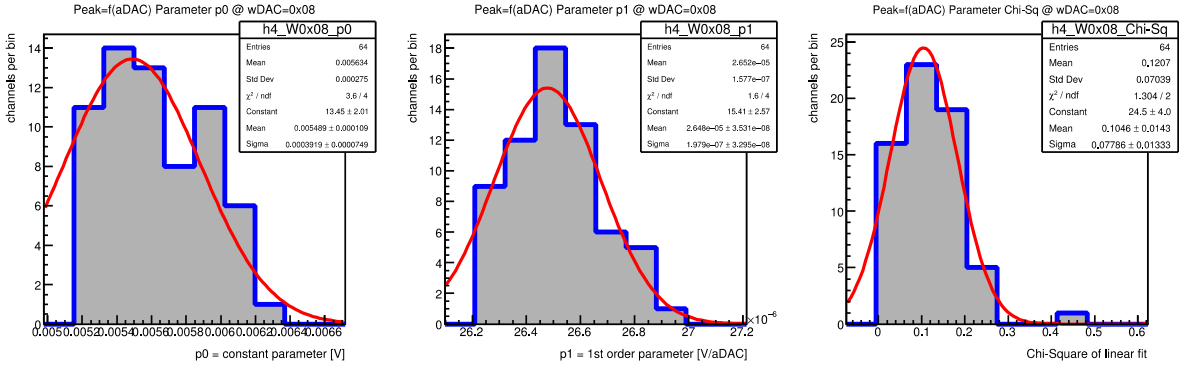


Figura 35 – Parâmetros de v_pico em função de $aDAC$ @ $wDAC = 0x08$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de carga calculada em função de $wDAC$:

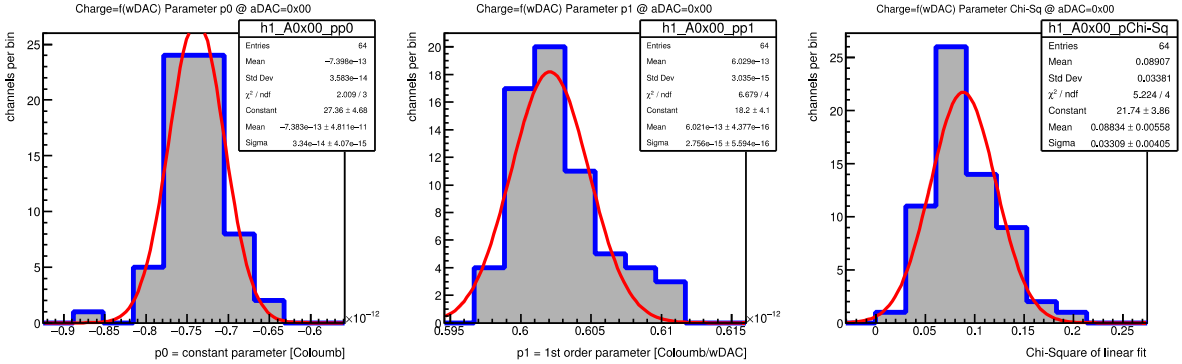


Figura 36 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x00$

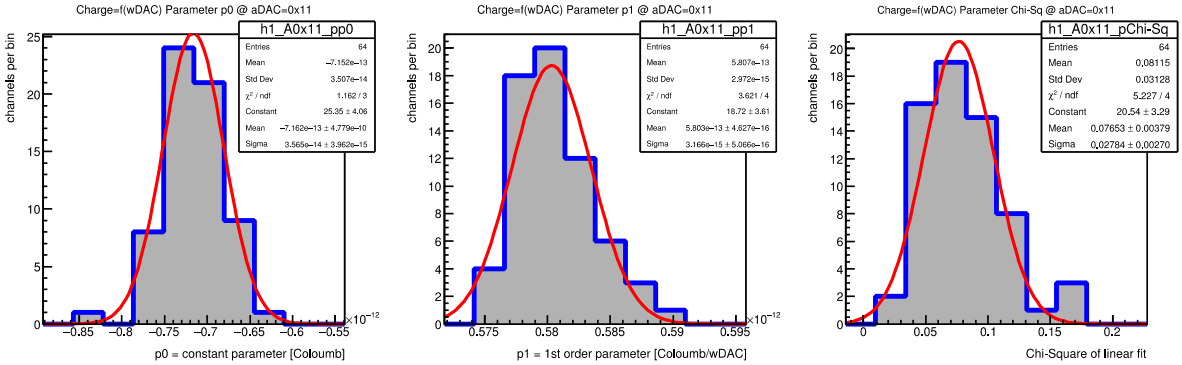


Figura 37 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x11$

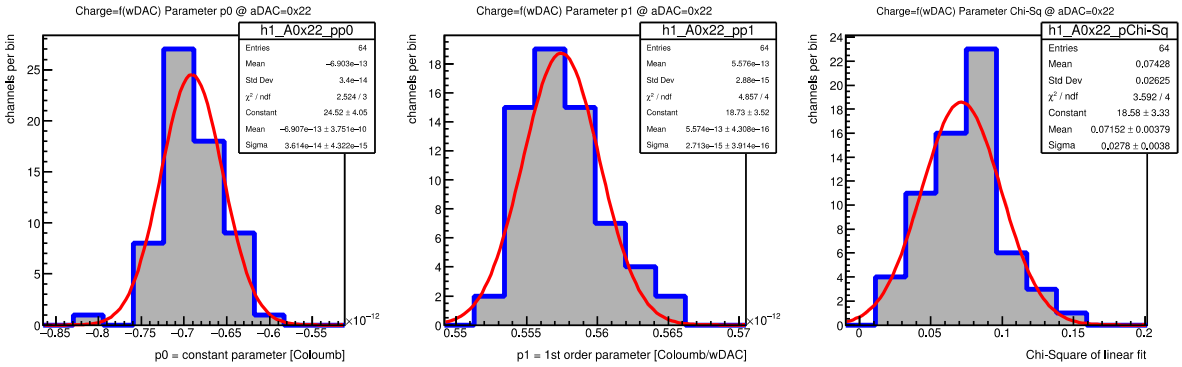


Figura 38 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x22$

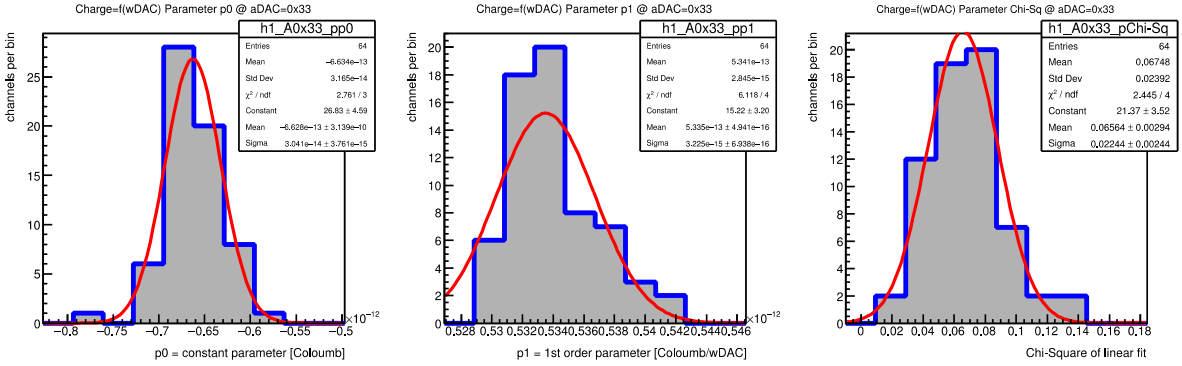


Figura 39 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x33$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de carga calculada em função de $wDAC$ (continuação):

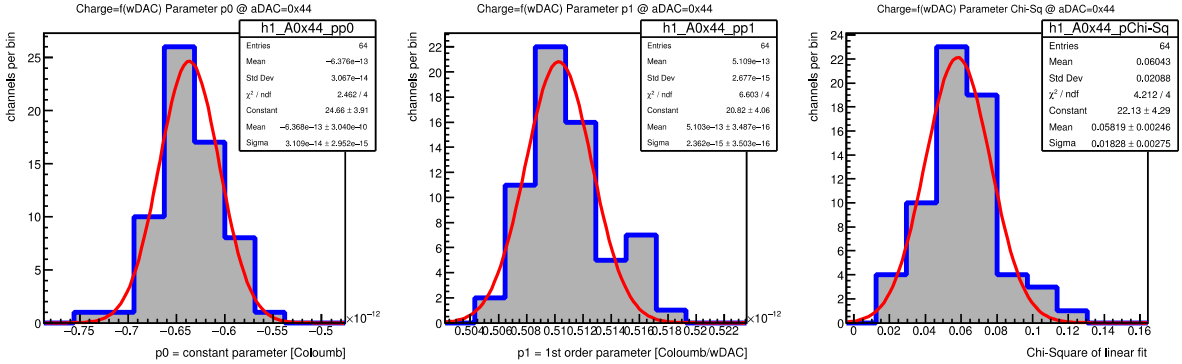


Figura 40 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x44$

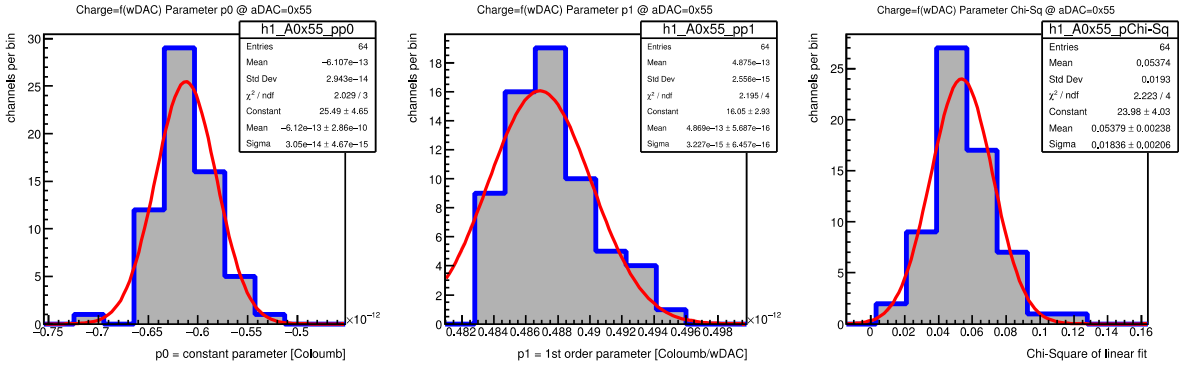


Figura 41 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x55$

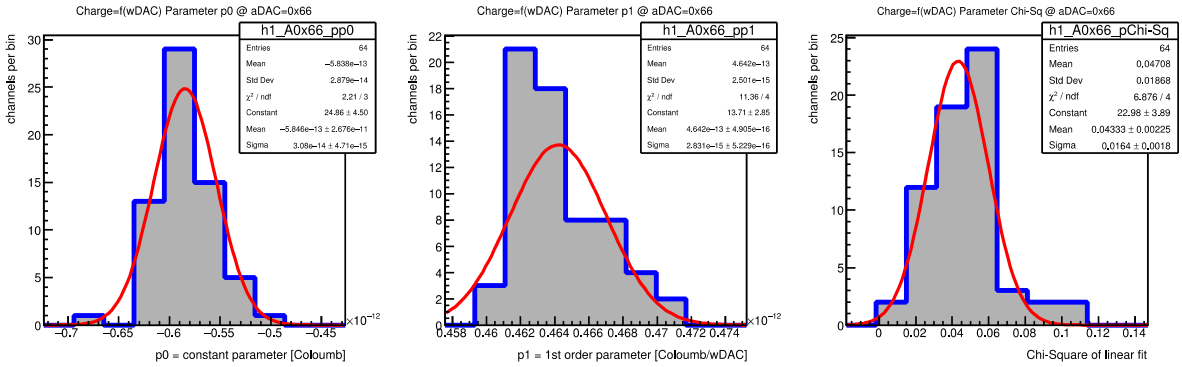


Figura 42 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x66$

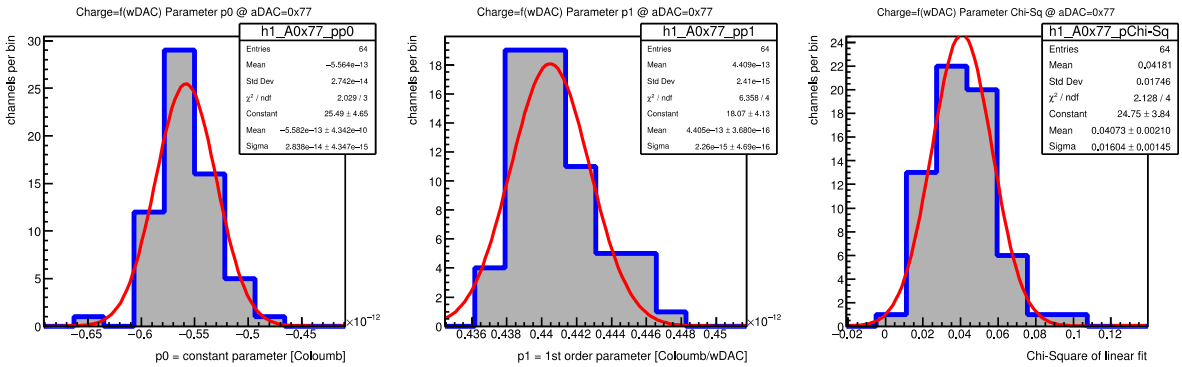


Figura 43 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x77$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de carga calculada em função de $wDAC$ (continuação):

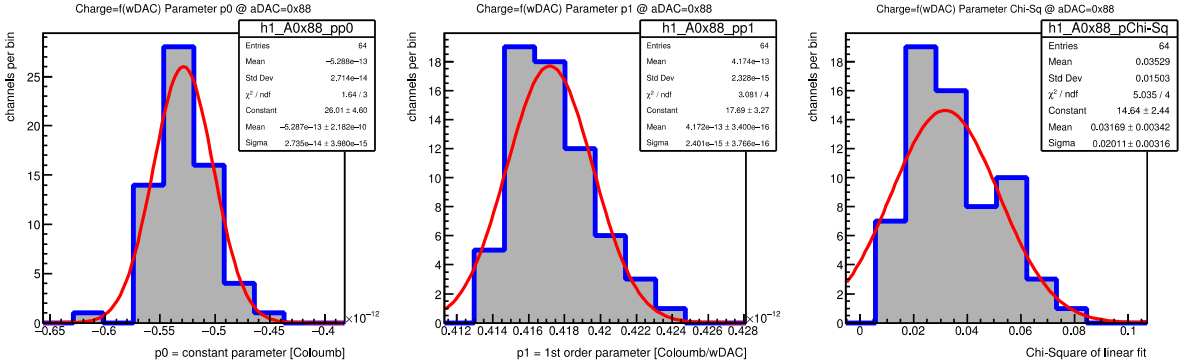


Figura 44 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x88$

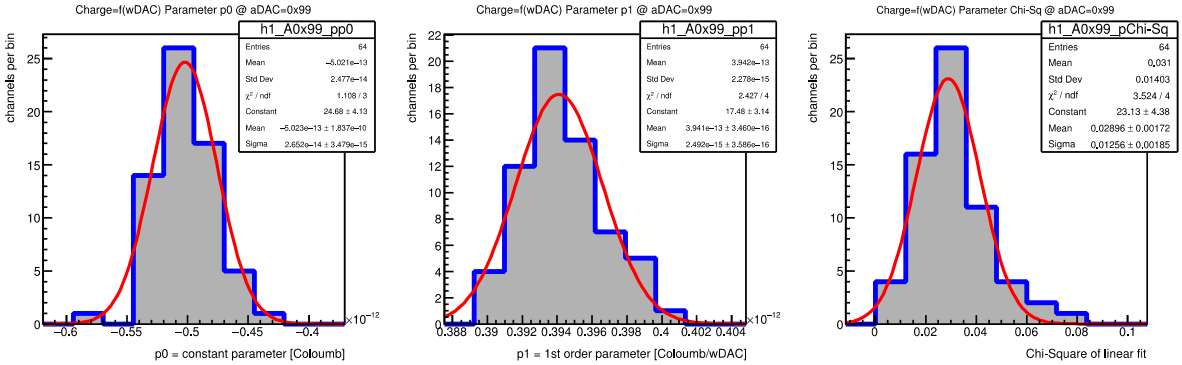


Figura 45 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x99$

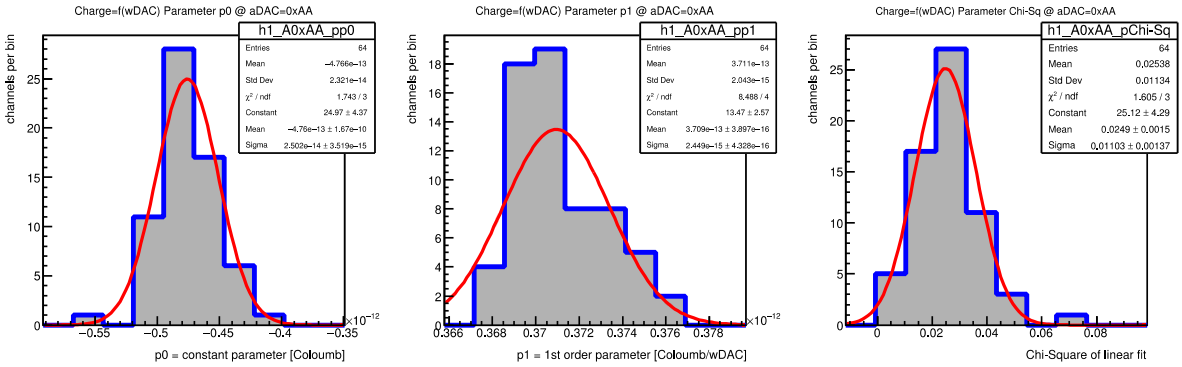


Figura 46 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0xAA$

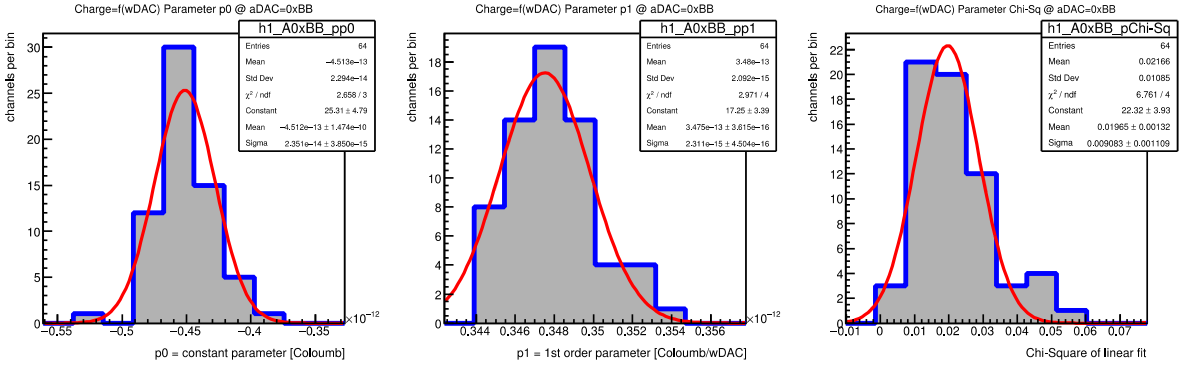


Figura 47 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0xBB$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de carga calculada em função de $wDAC$ (continuação):

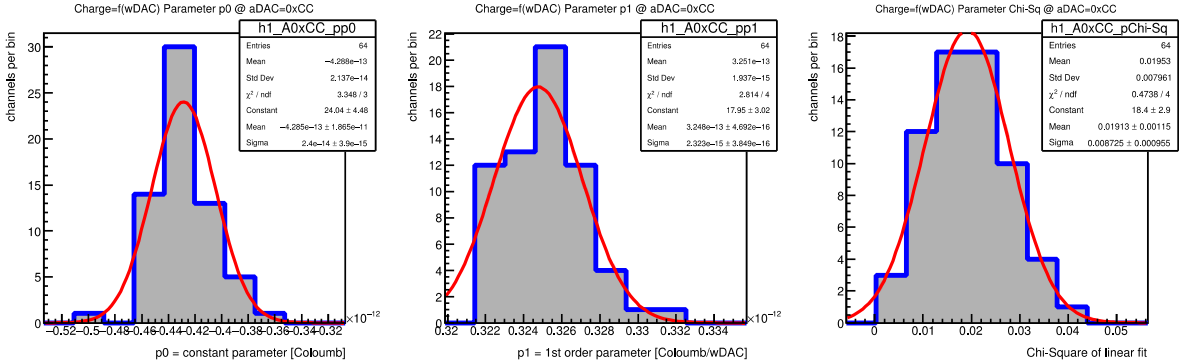


Figura 48 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xCC

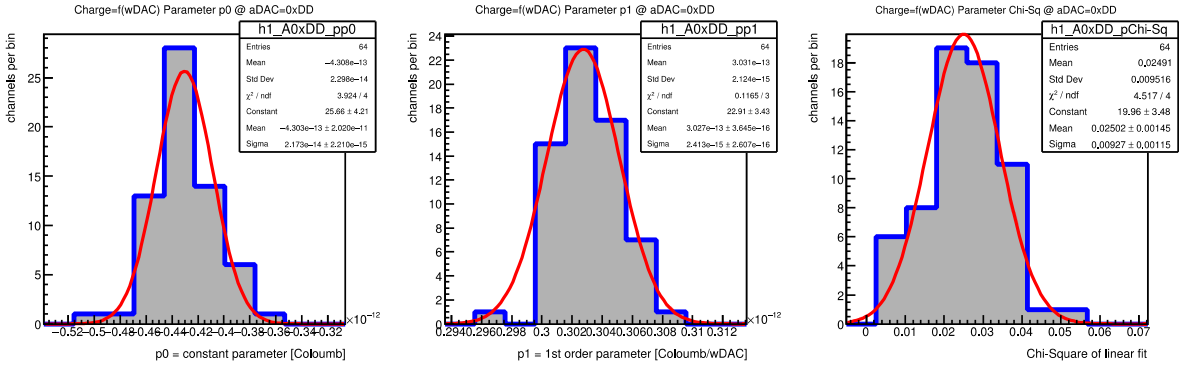


Figura 49 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xDD

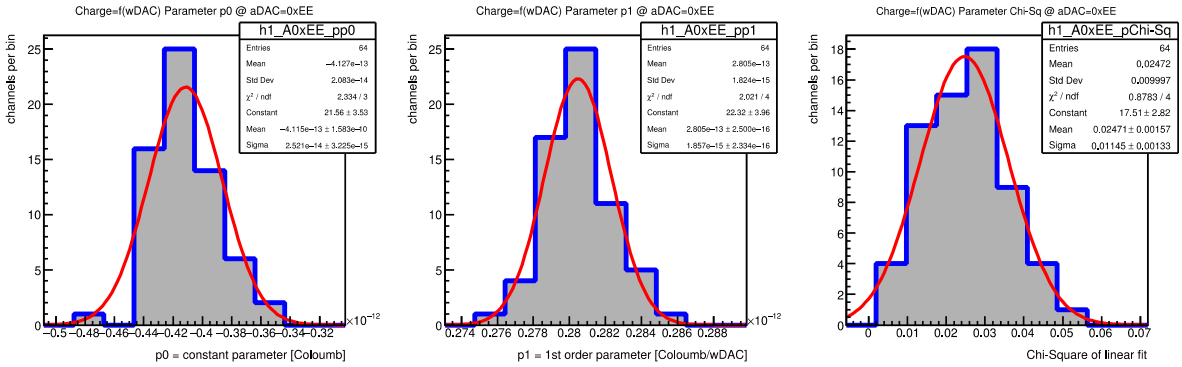


Figura 50 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xEE

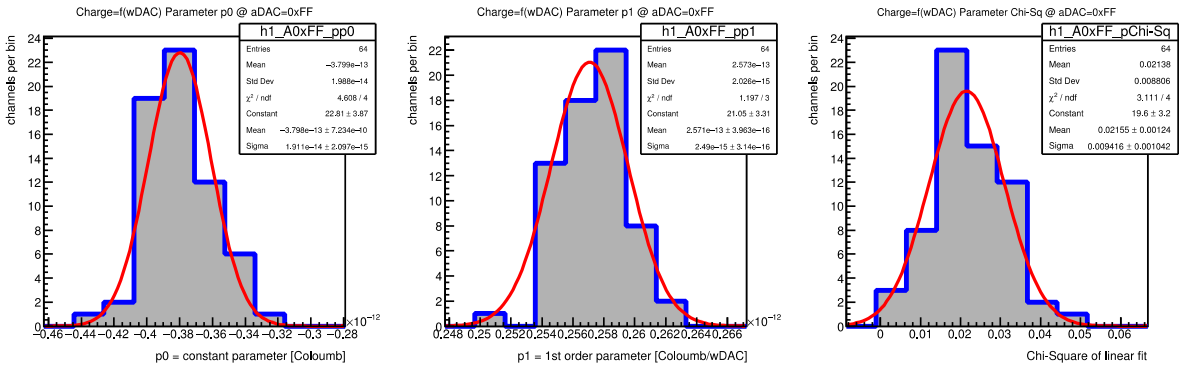


Figura 51 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xFF

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de área corrigida em função de $wDAC$:

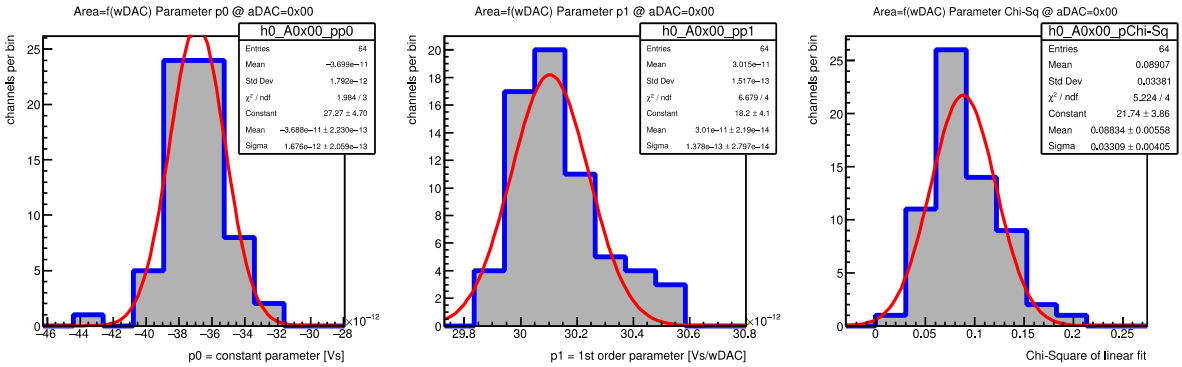


Figura 52 – Parâmetros de área em função de $wDAC$ @ $aDAC = 0x00$

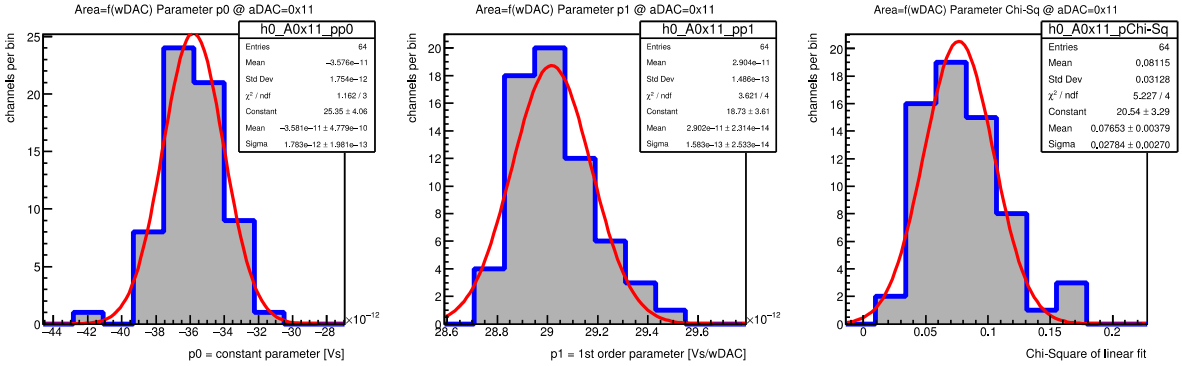


Figura 53 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x11$

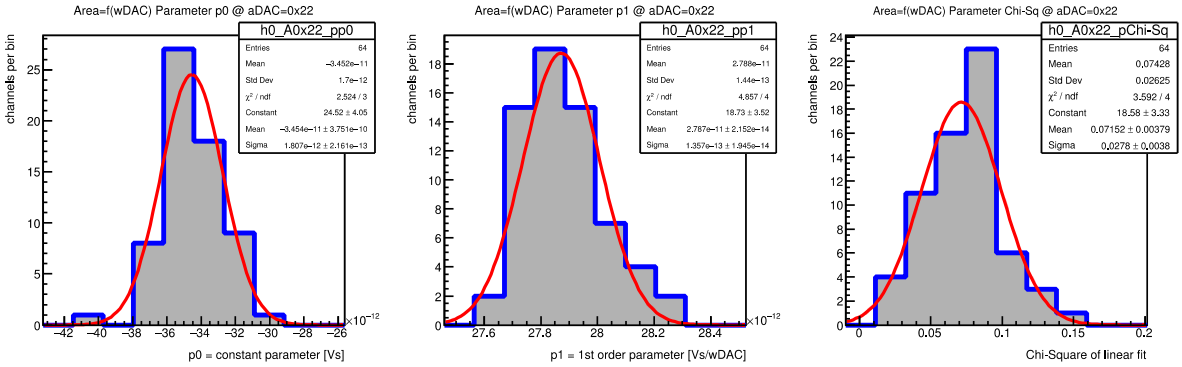


Figura 54 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x22$

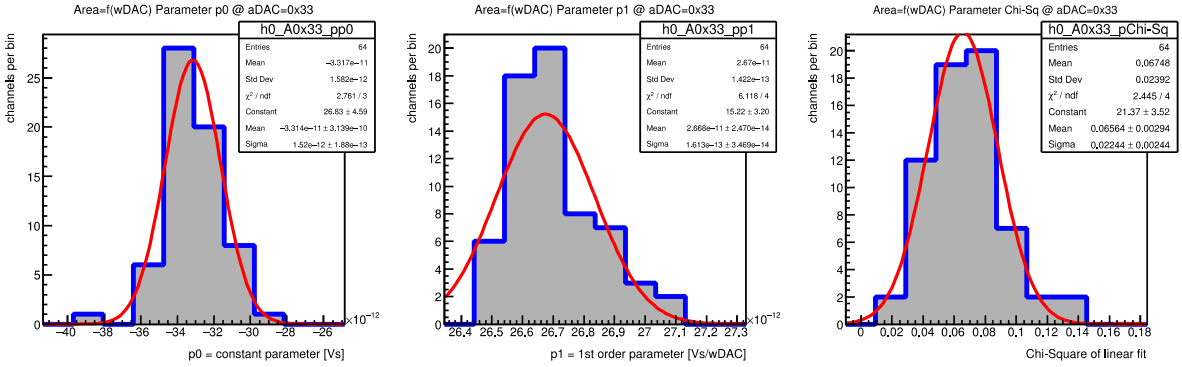


Figura 55 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x33$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de área corrigida em função de $wDAC$ (continuação):

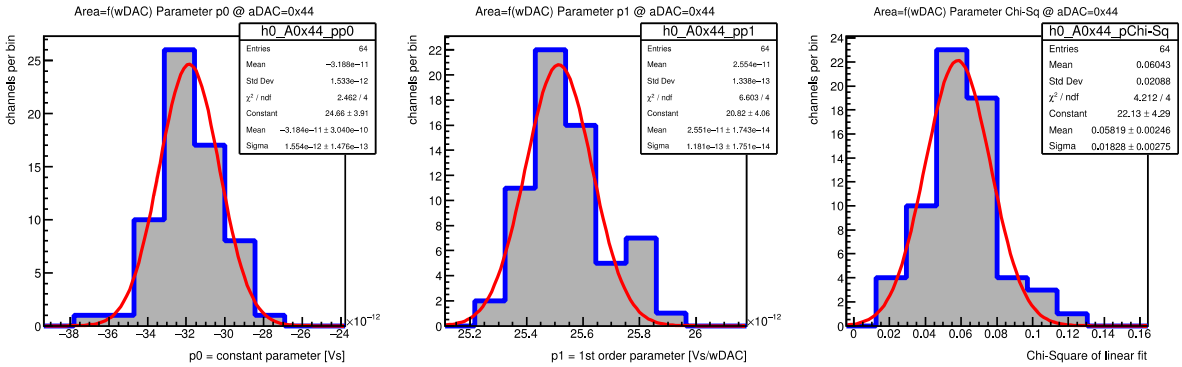


Figura 56 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x44$

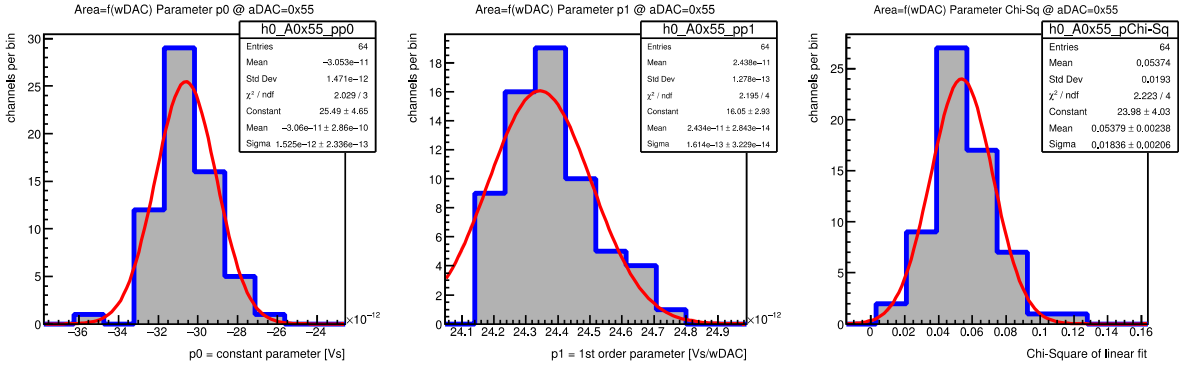


Figura 57 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x55$

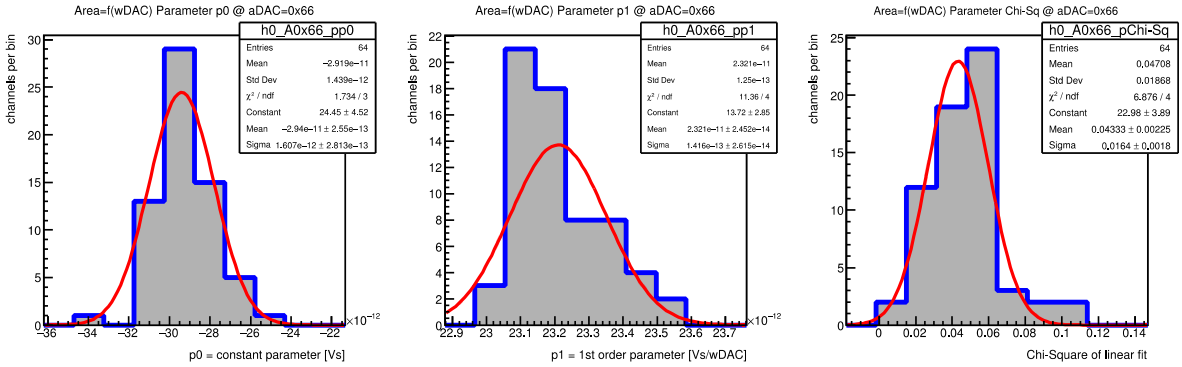


Figura 58 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x66$

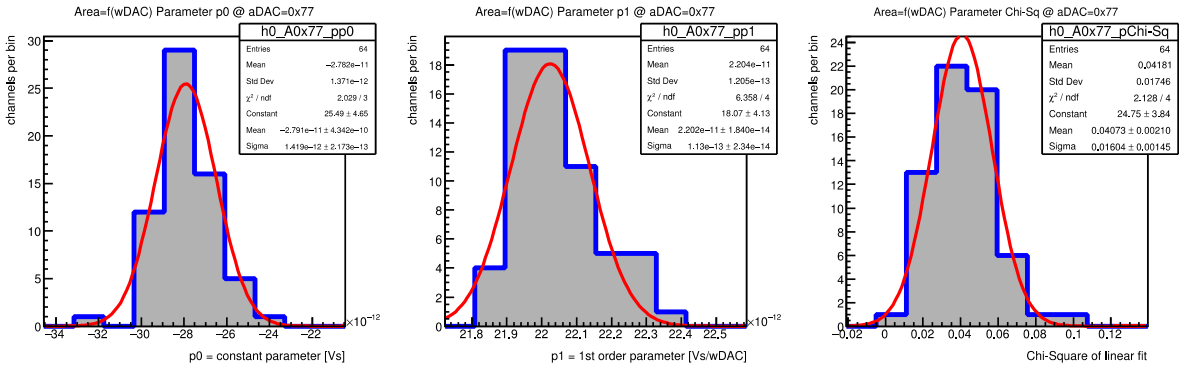


Figura 59 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0x77$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de área corrigida em função de $wDAC$ (continuação):

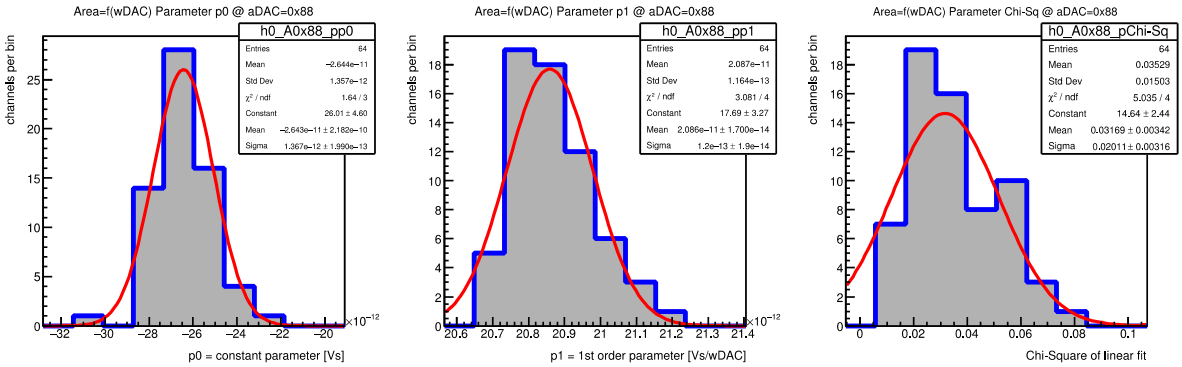


Figura 60 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$x88

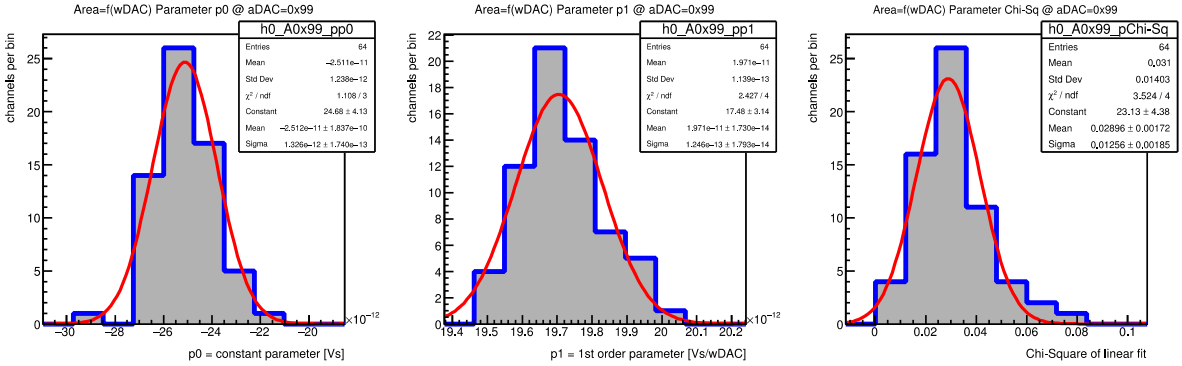


Figura 61 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$x99

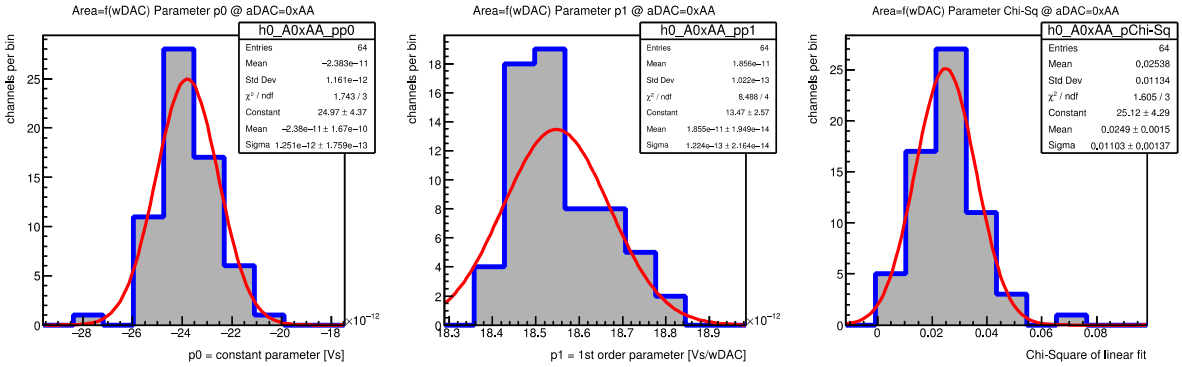


Figura 62 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xAA

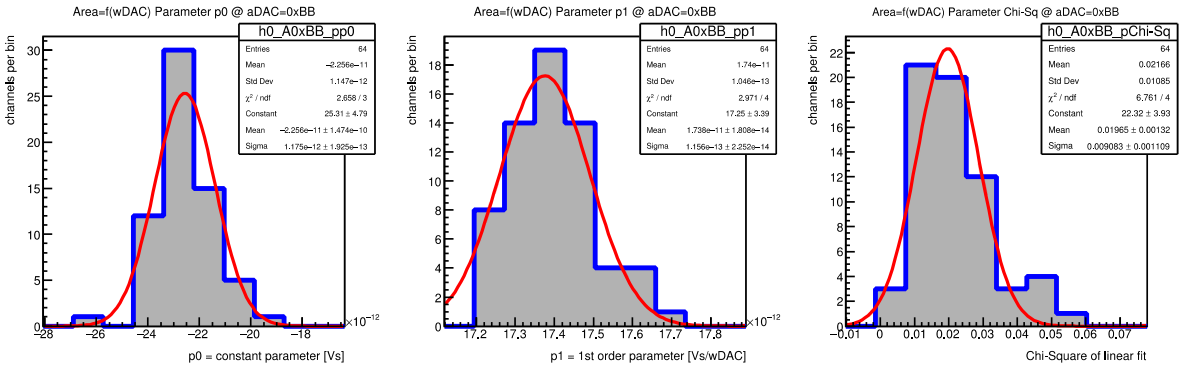


Figura 63 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xBB

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de área corrigida em função de $wDAC$ (continuação):

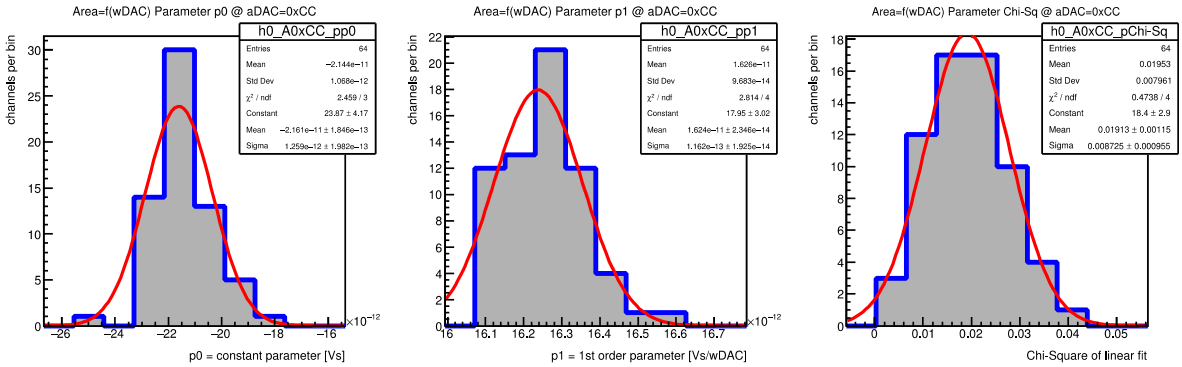


Figura 64 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xCC

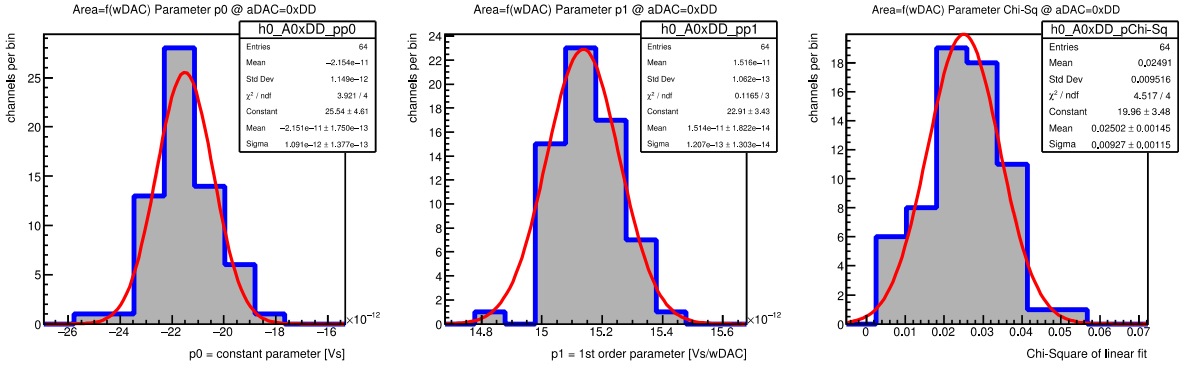


Figura 65 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xDD

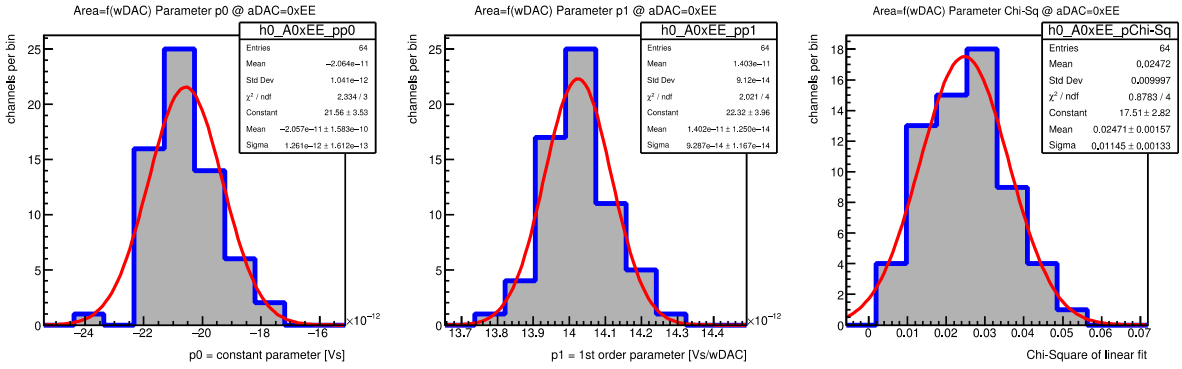


Figura 66 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xEE

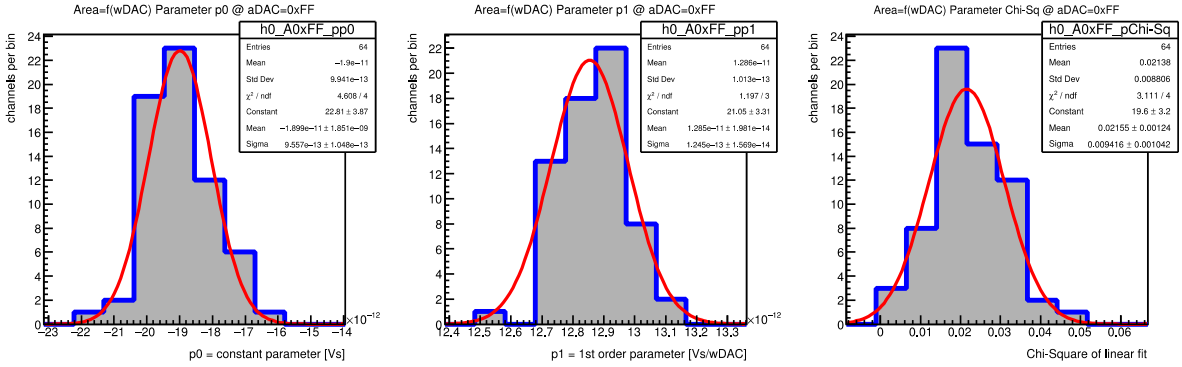


Figura 67 – Parâmetros de carga em função de $wDAC$ @ $aDAC = 0$xFF

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de largura do pulso em função de $wDAC$:

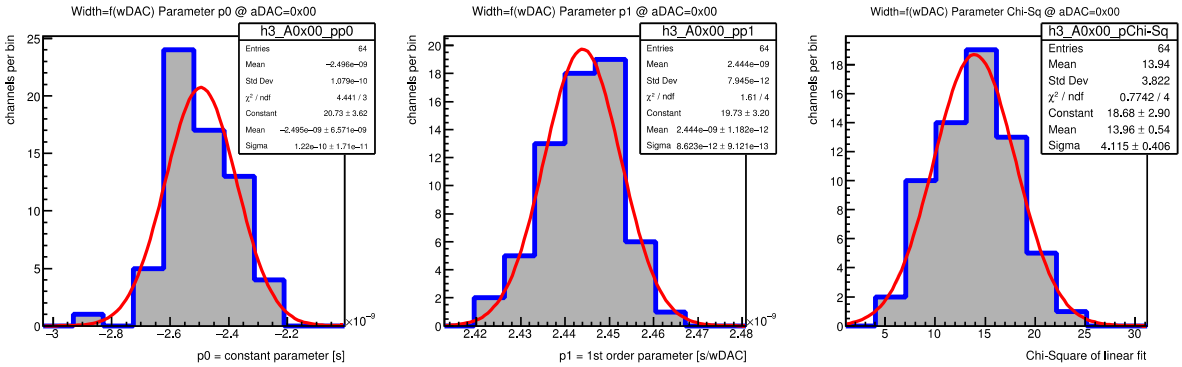


Figura 68 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x00$

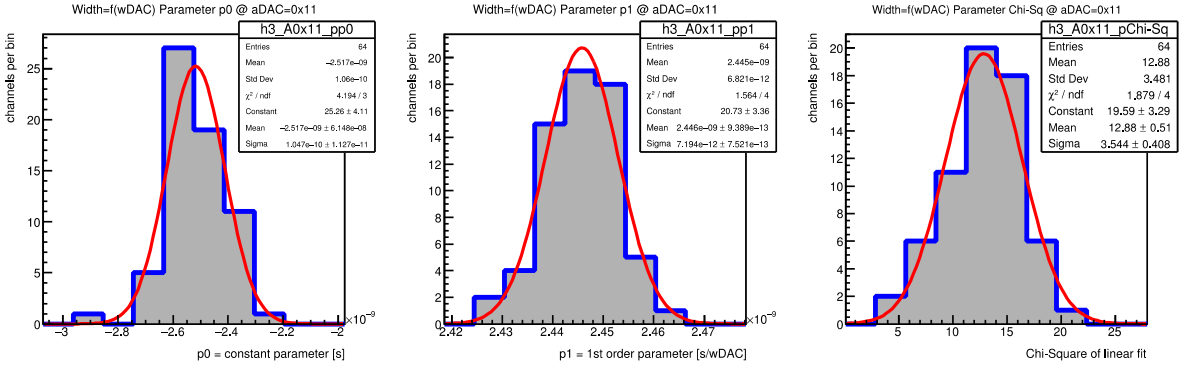


Figura 69 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x11$

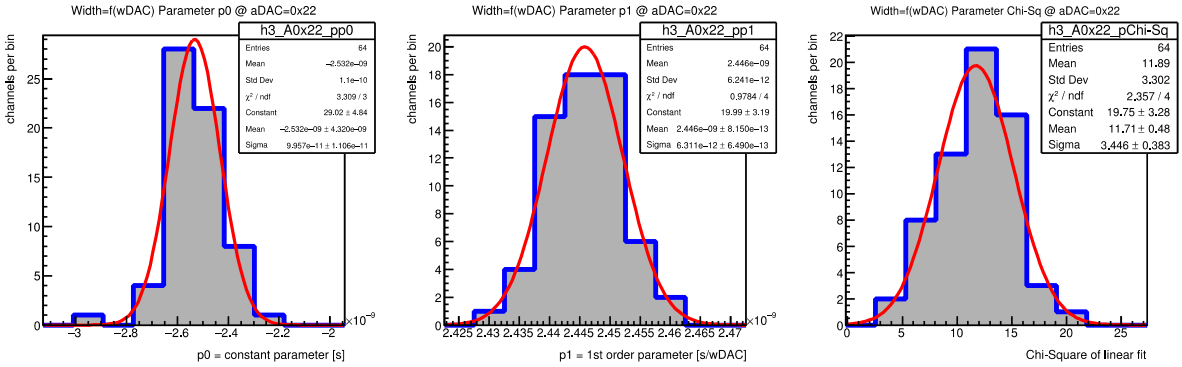


Figura 70 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x22$

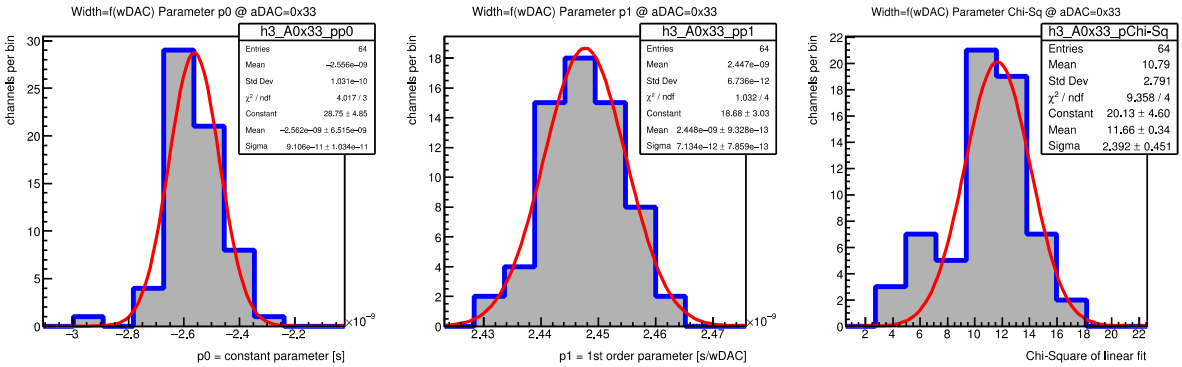


Figura 71 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x33$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de largura do pulso em função de $wDAC$ (continuação):

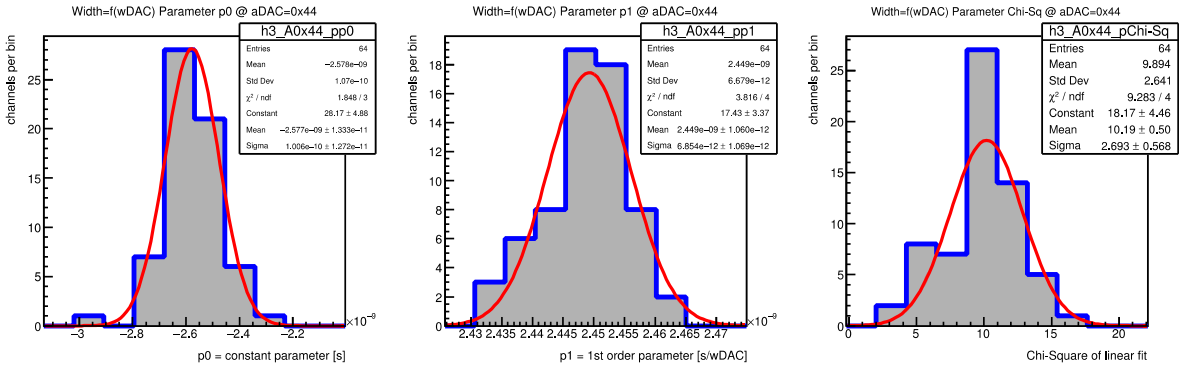


Figura 72 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x44$

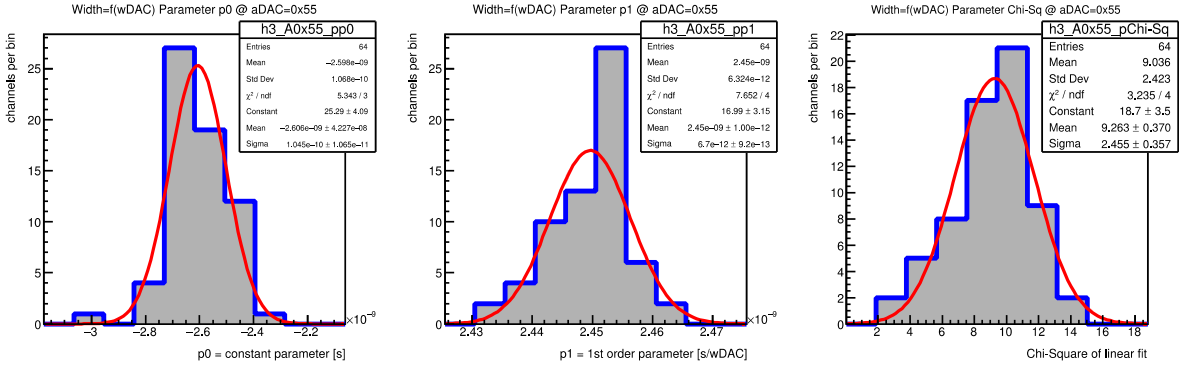


Figura 73 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x55$

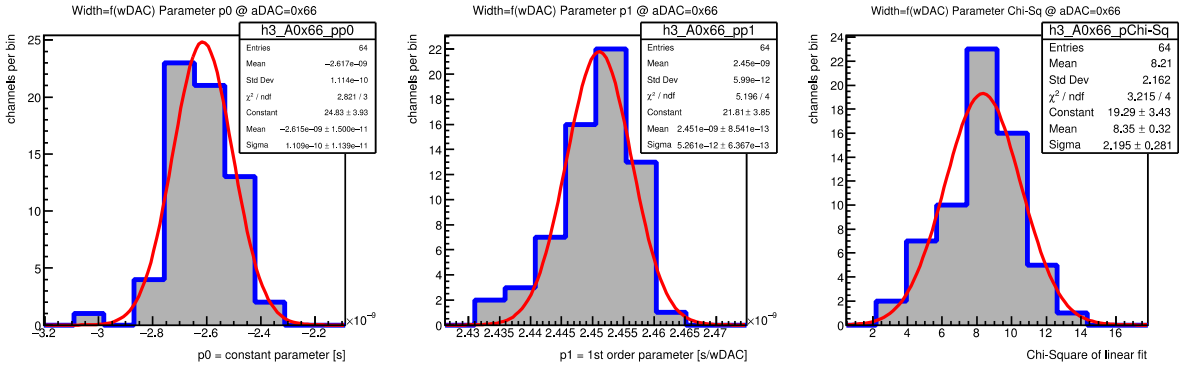


Figura 74 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x66$

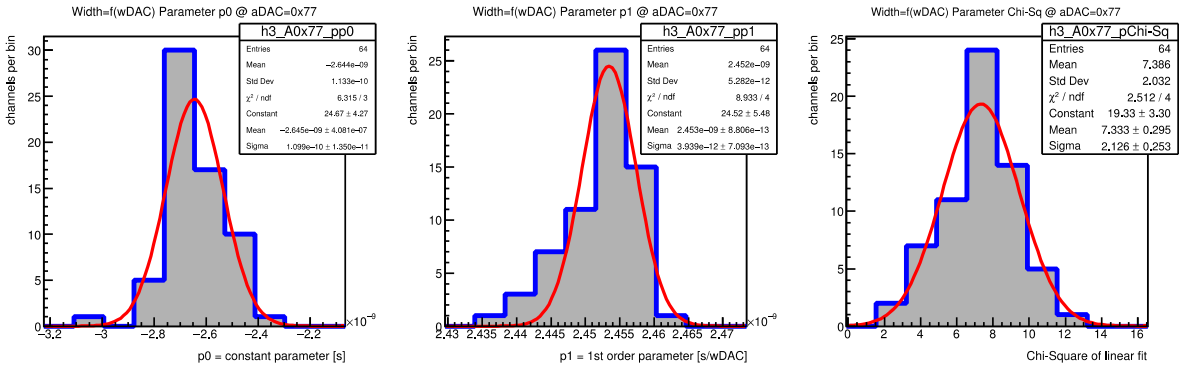


Figura 75 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x77$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de largura do pulso em função de $wDAC$ (continuação):

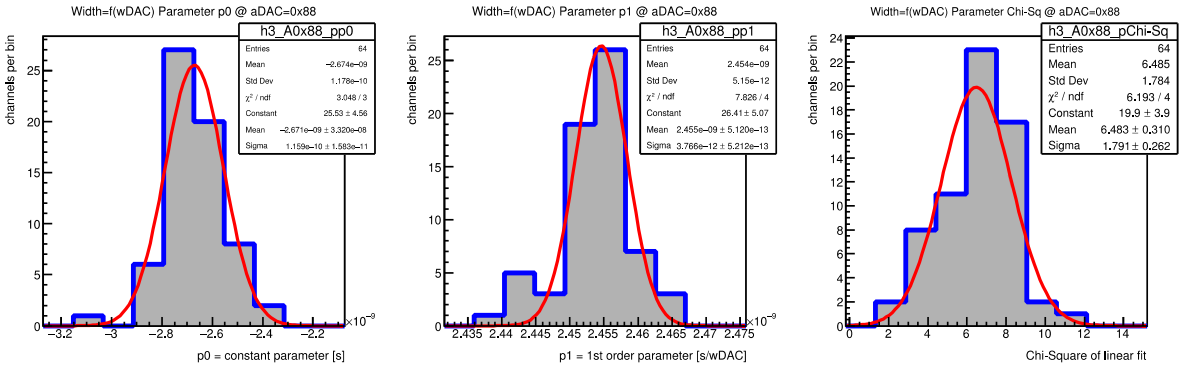


Figura 76 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x88$

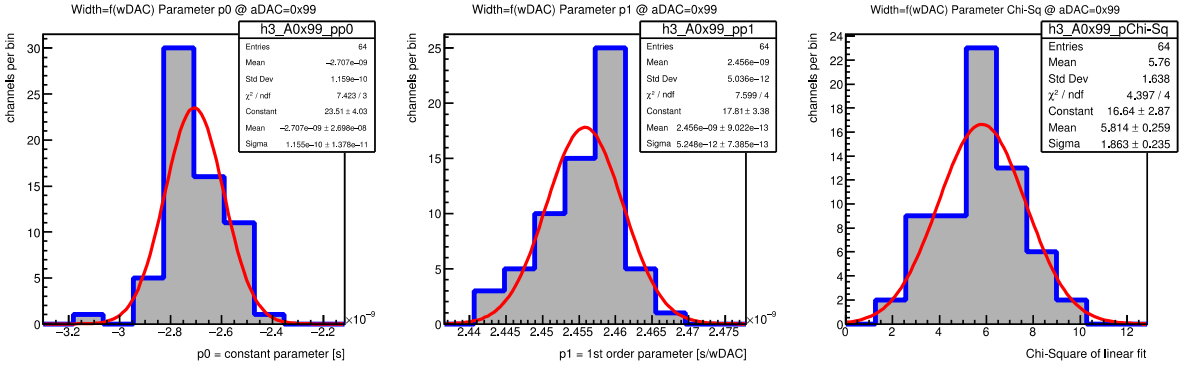


Figura 77 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0x99$

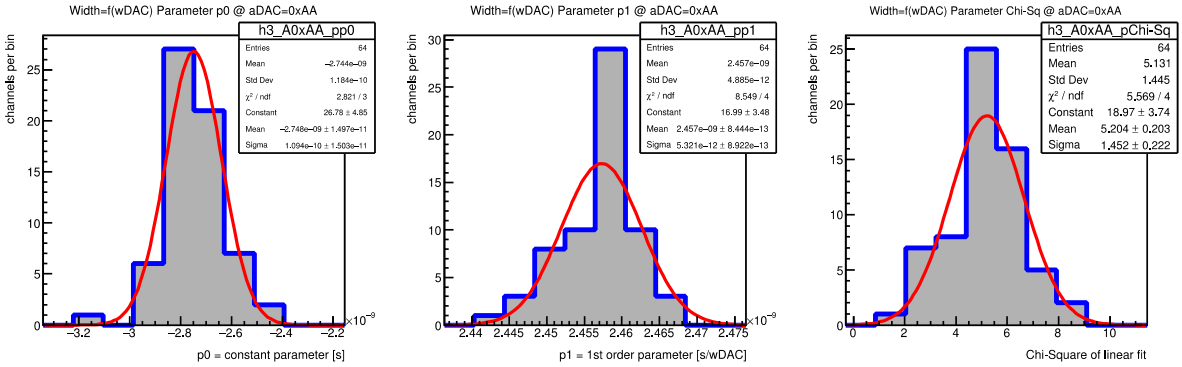


Figura 78 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0xAA$

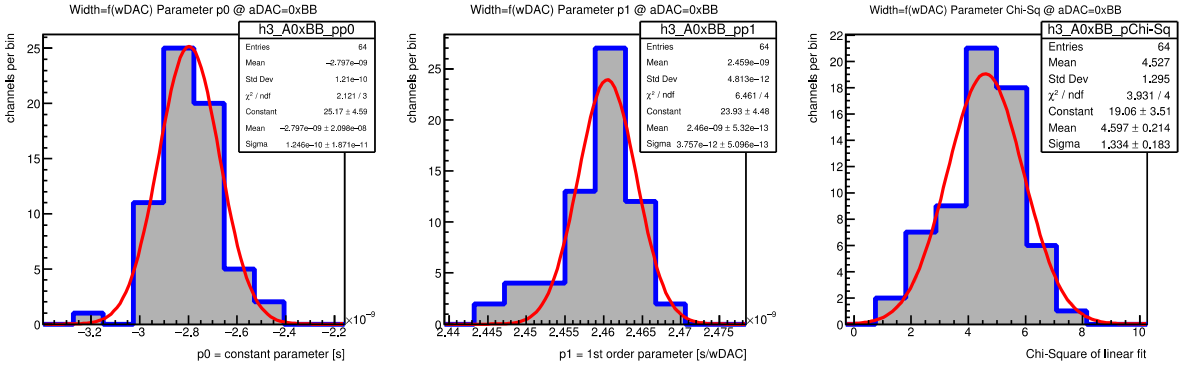


Figura 79 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0xBB$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de largura do pulso em função de $wDAC$ (continuação):

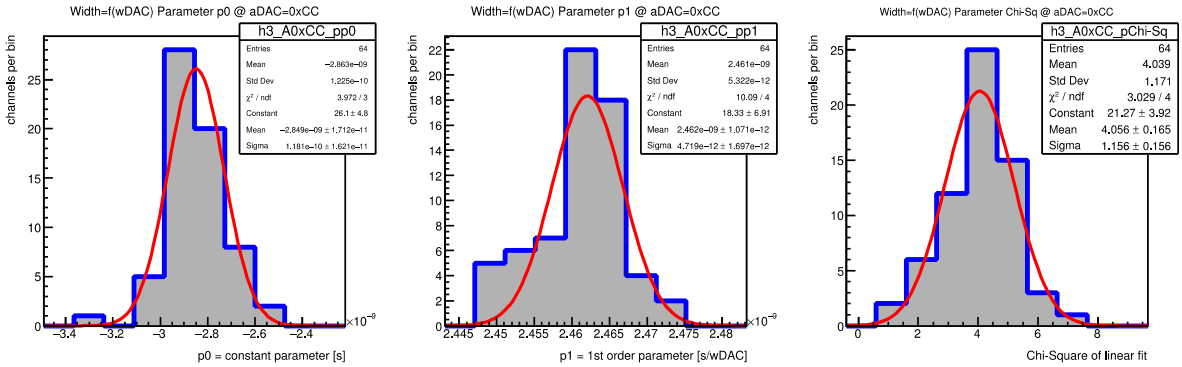


Figura 80 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xCC

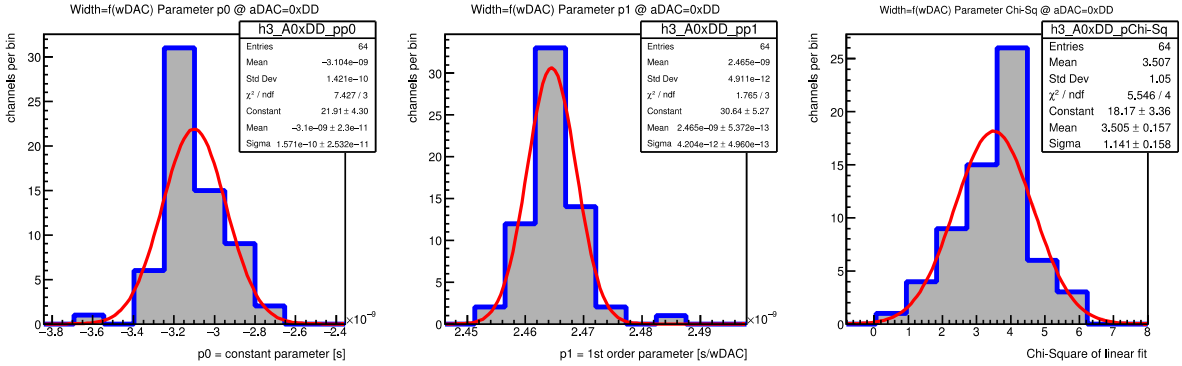


Figura 81 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xDD

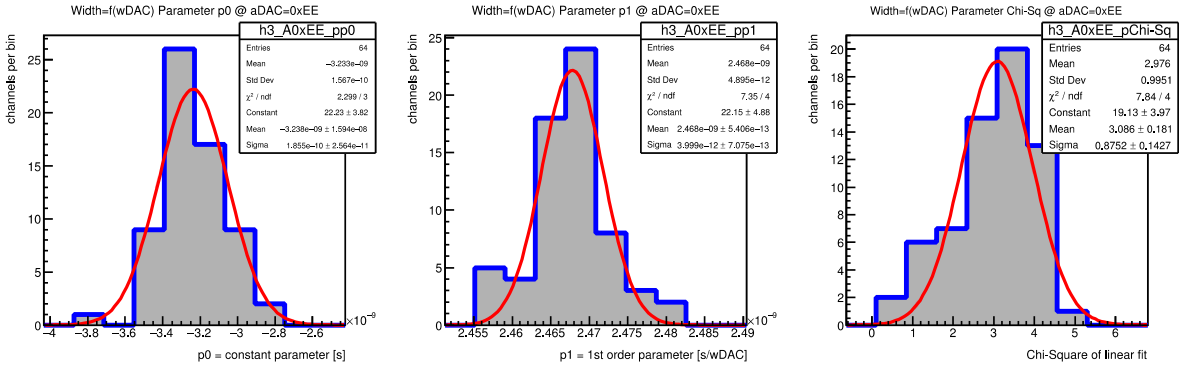


Figura 82 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xEE

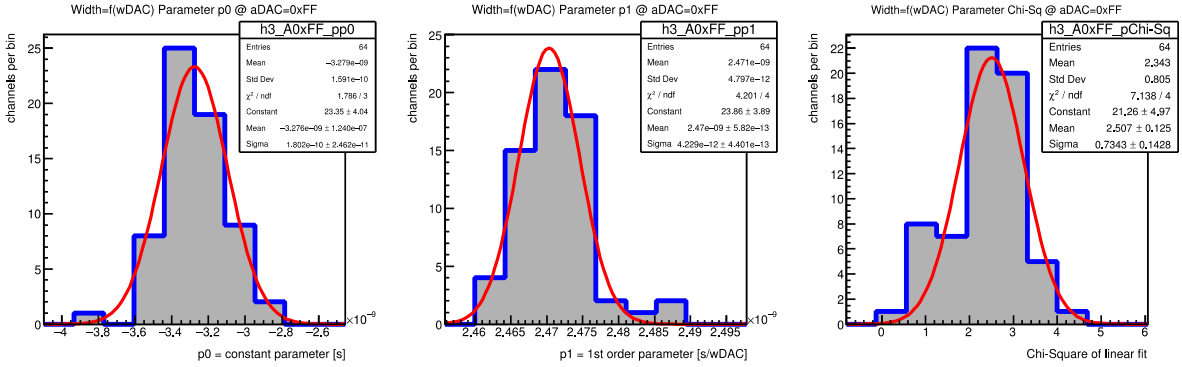


Figura 83 – Parâmetros de largura em função de $wDAC$ @ $aDAC = 0$xFF

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de temporização em função de $wDAC$:

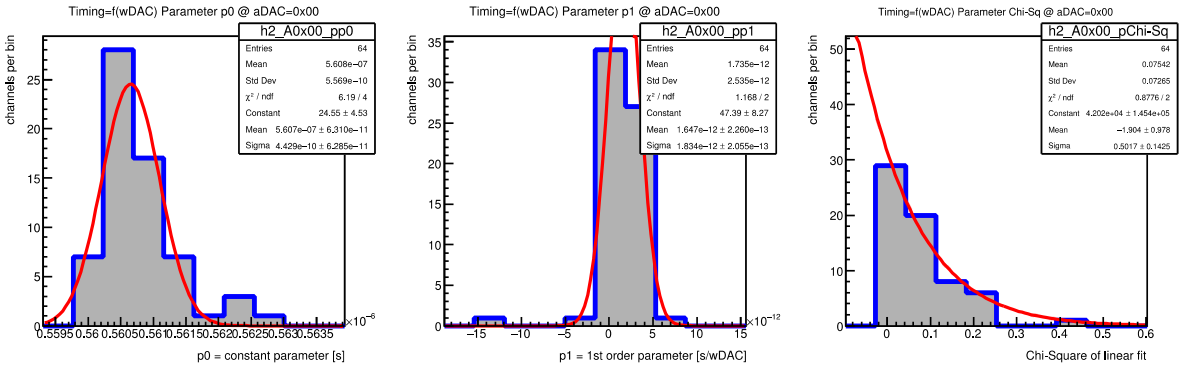


Figura 84 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0x00$

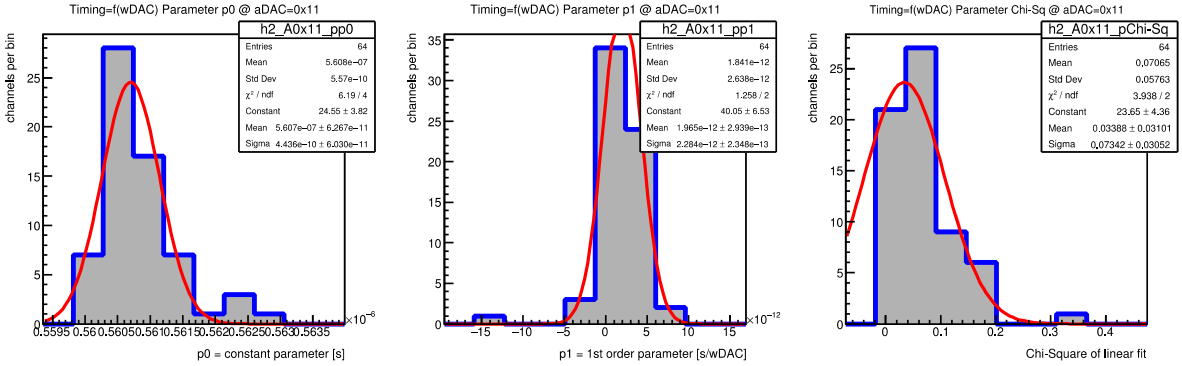


Figura 85 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0x11$

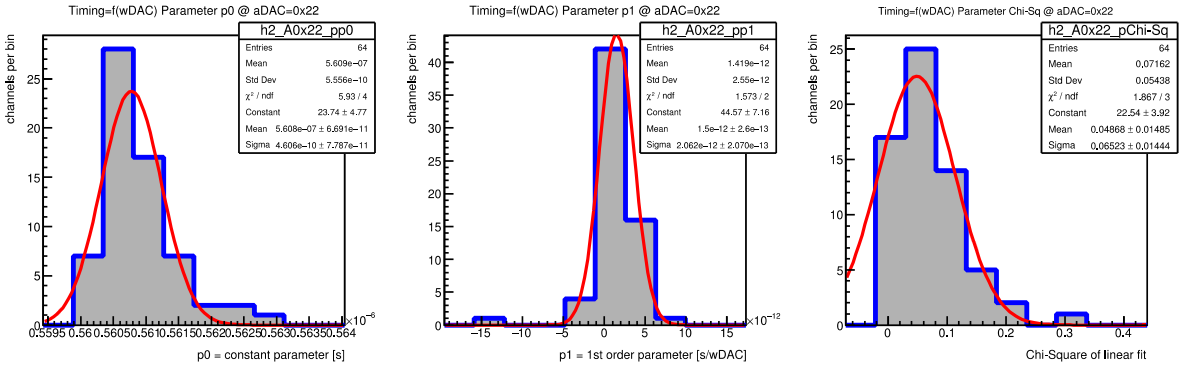


Figura 86 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0x22$

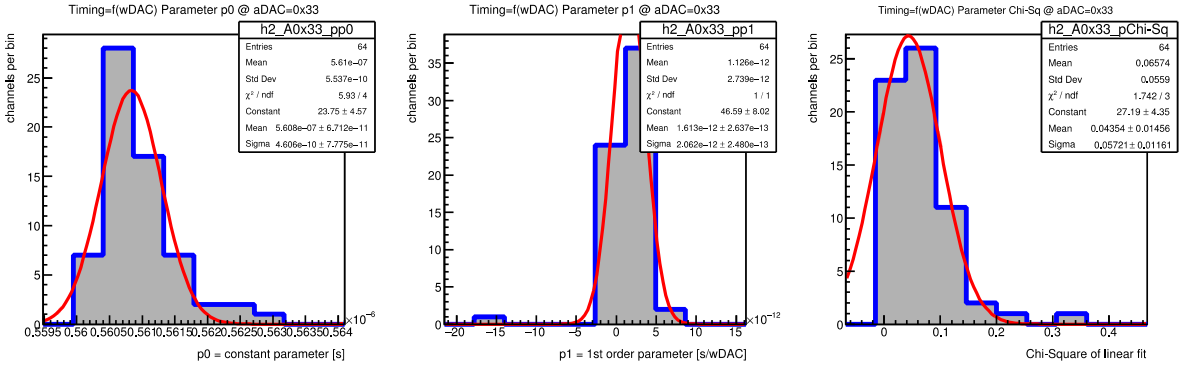


Figura 87 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0x33$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de temporização em função de $wDAC$ (continuação):

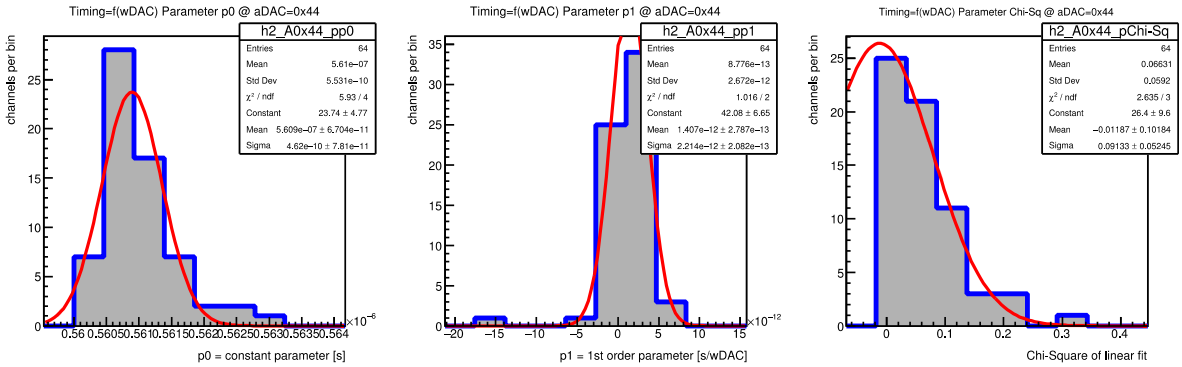


Figura 88 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x44

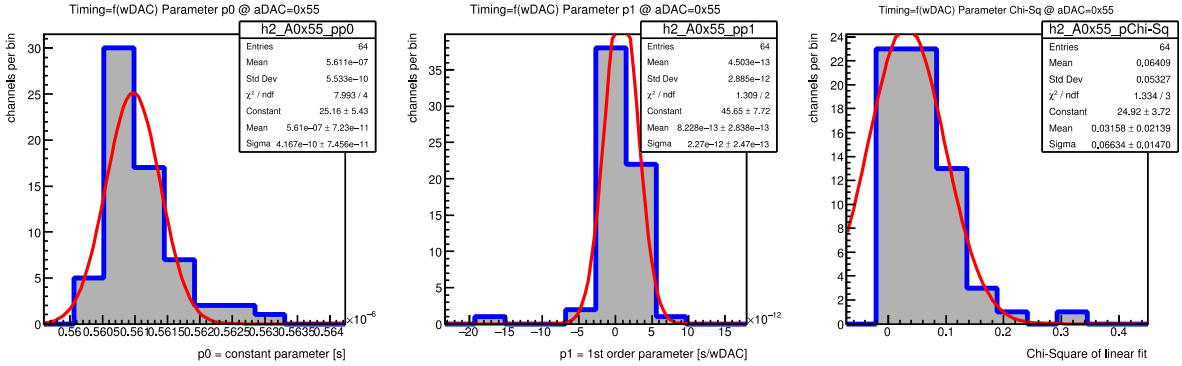


Figura 89 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x55

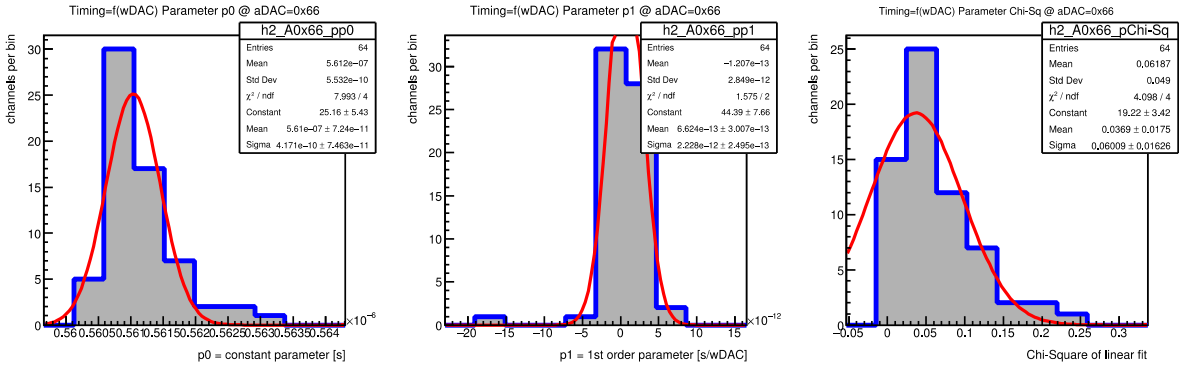


Figura 90 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x66

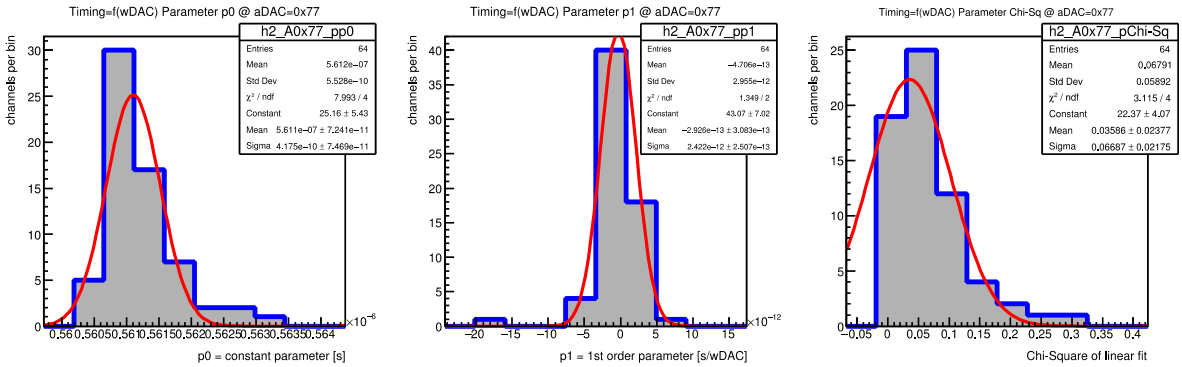


Figura 91 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$x77

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de temporização em função de $wDAC$ (continuação):

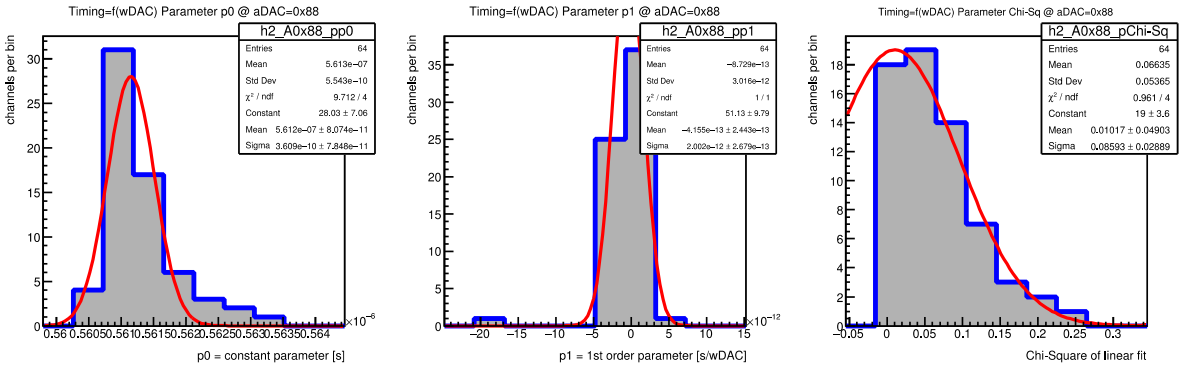


Figura 92 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0x88$

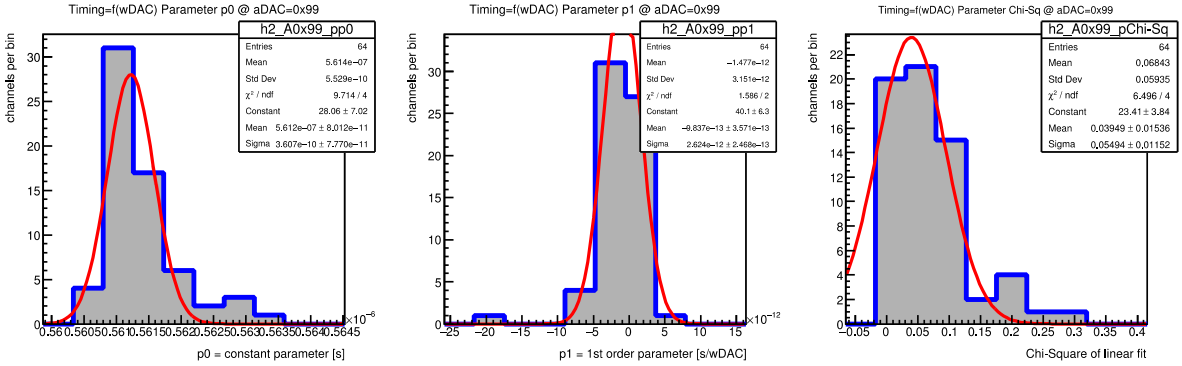


Figura 93 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0x99$

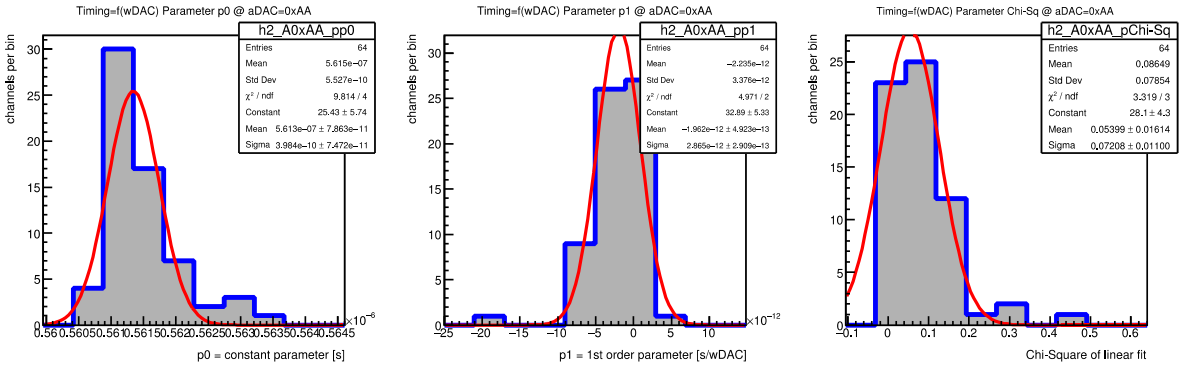


Figura 94 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0xAA$

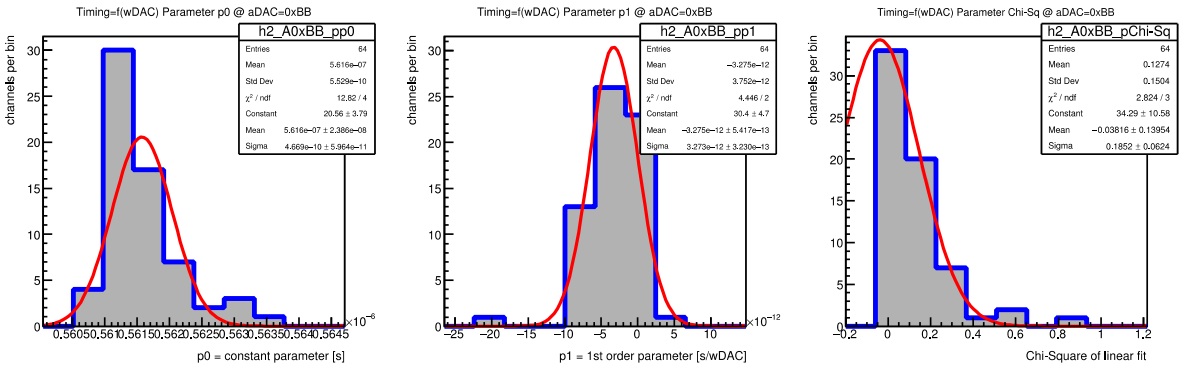


Figura 95 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0xBB$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de temporização em função de $wDAC$ (continuação):

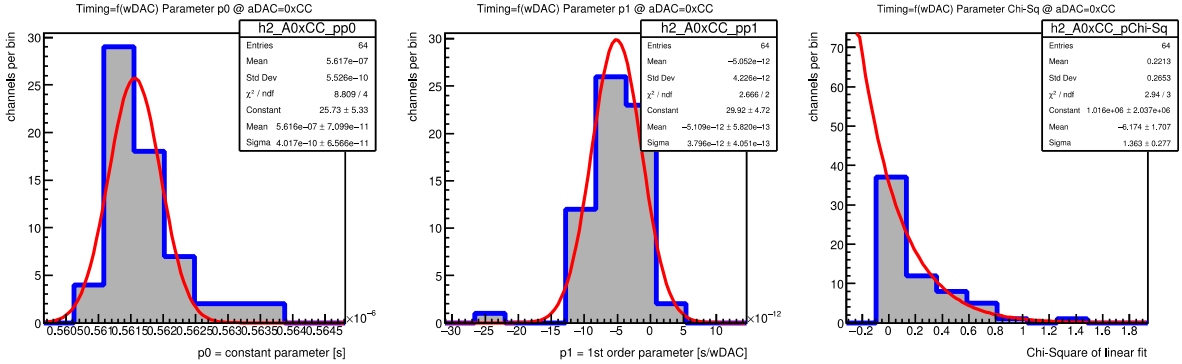


Figura 96 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xCC

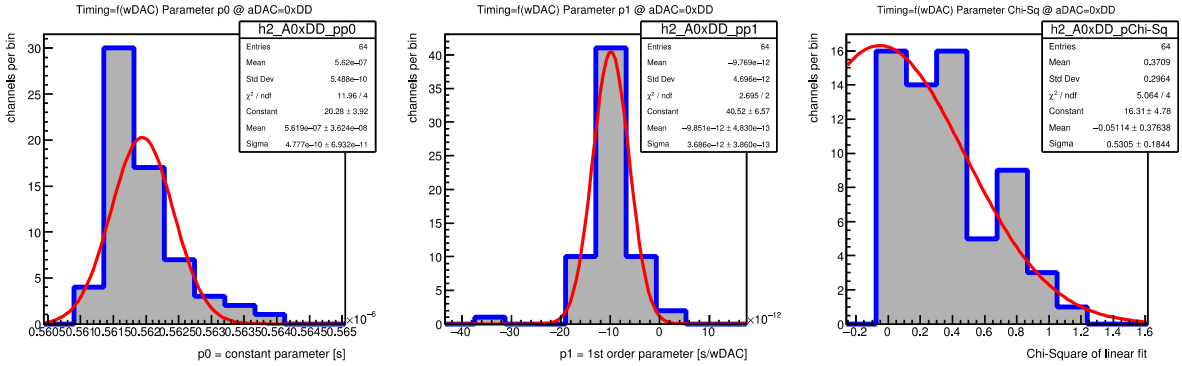


Figura 97 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xDD

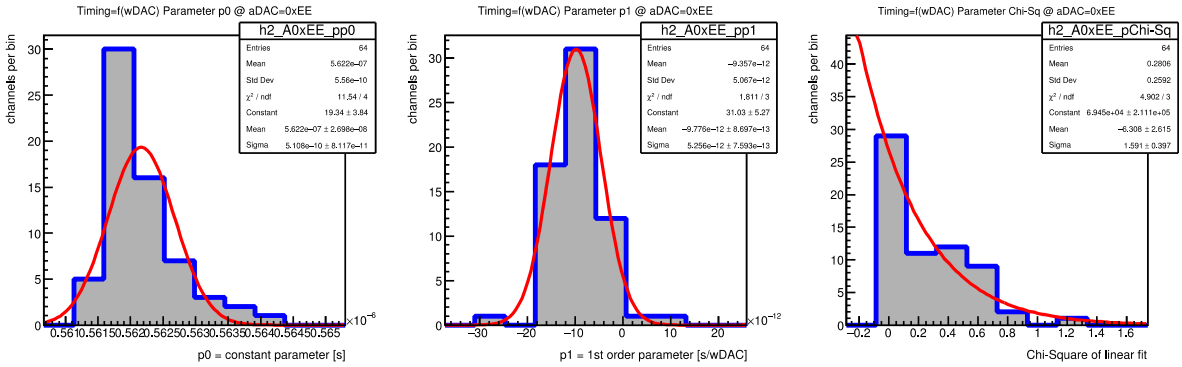


Figura 98 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xEE

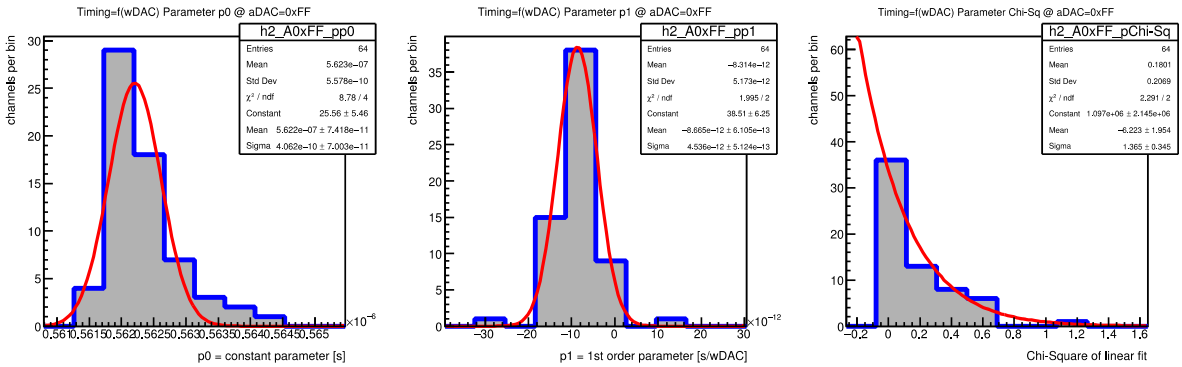


Figura 99 – Parâmetros de temporização em função de $wDAC$ @ $aDAC = 0$xFF

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de tensão de pico em função de $wDAC$:

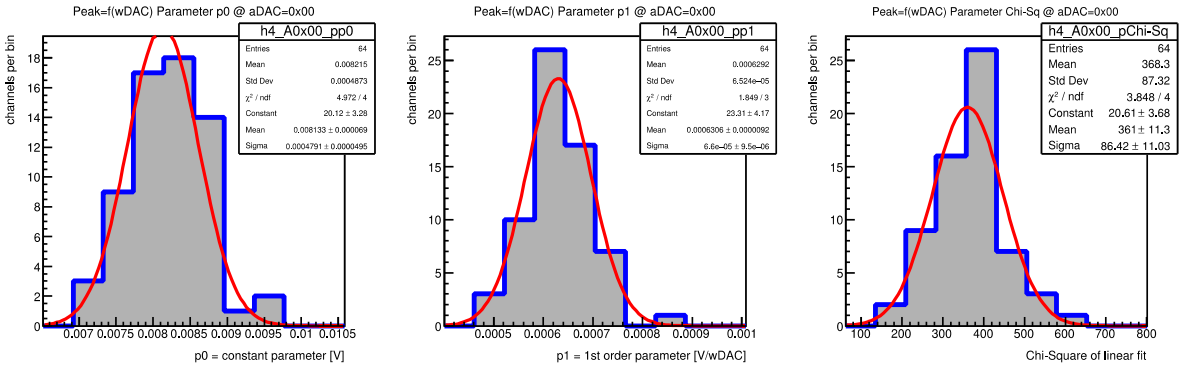


Figura 100 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0x00$

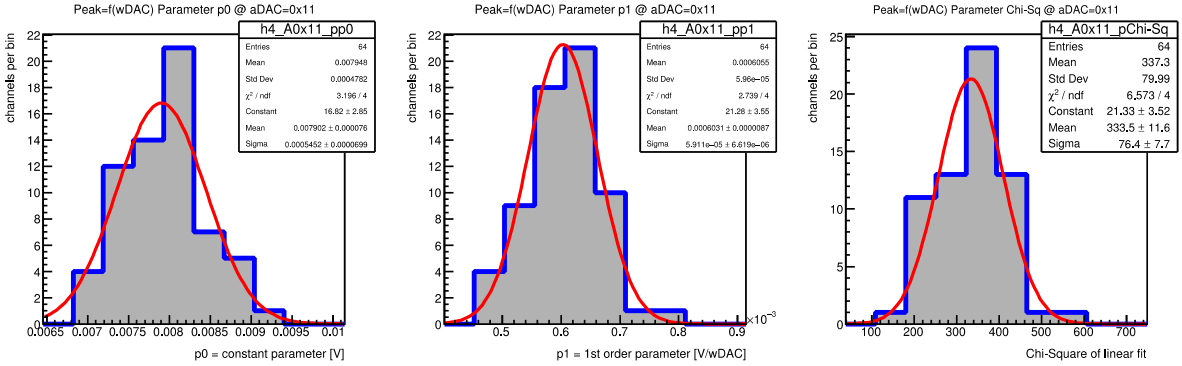


Figura 101 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0x11$

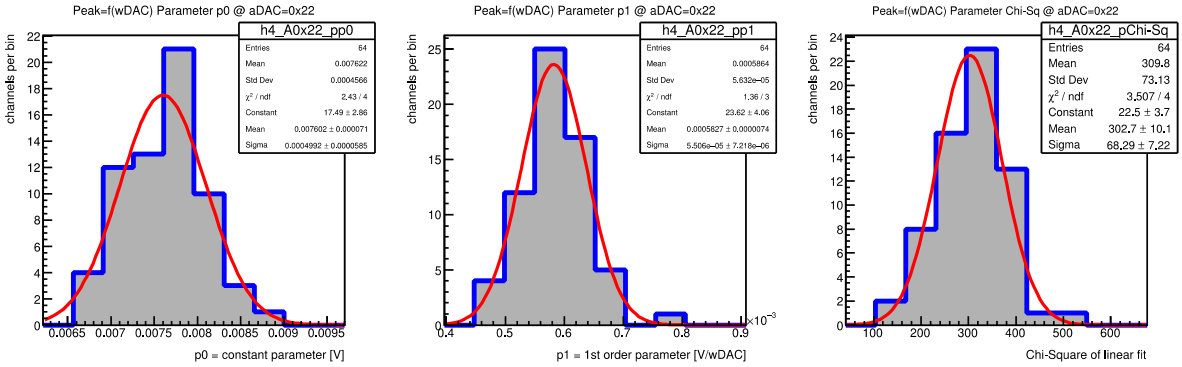


Figura 102 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0x22$

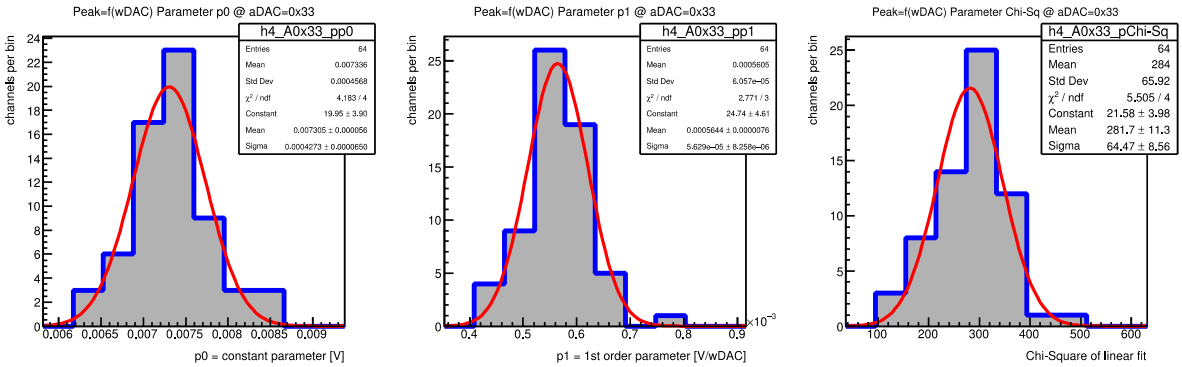


Figura 103 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0x33$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de tensão de pico em função de $wDAC$ (continuação):

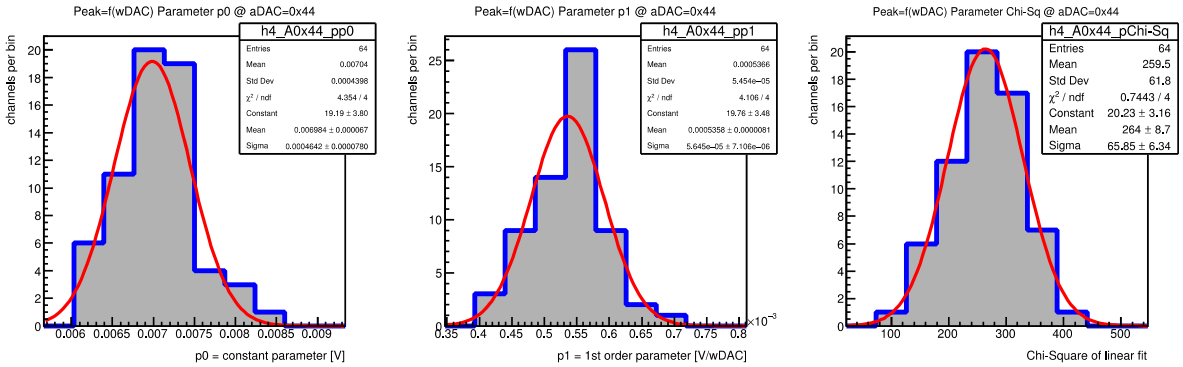


Figura 104 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x44

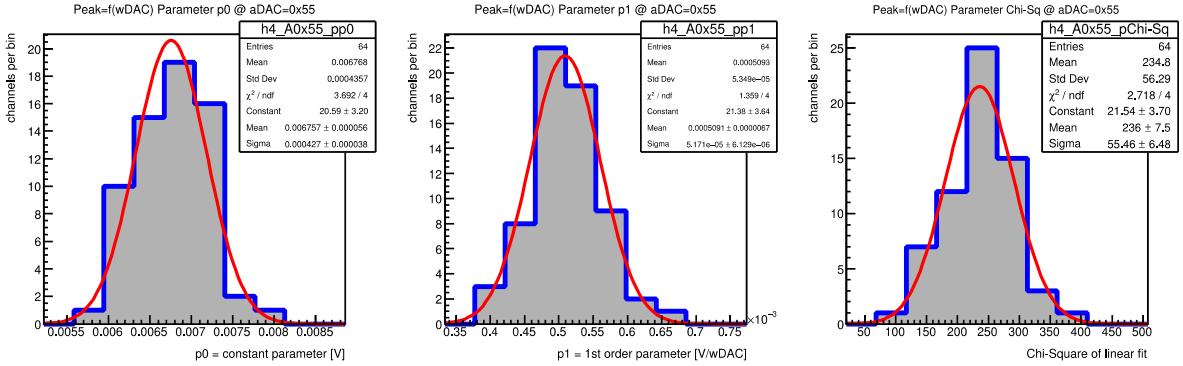


Figura 105 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x55

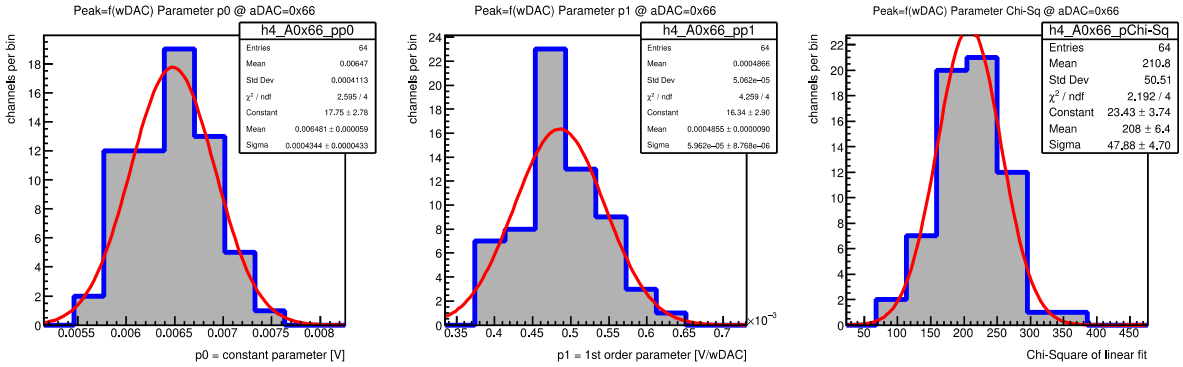


Figura 106 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x66

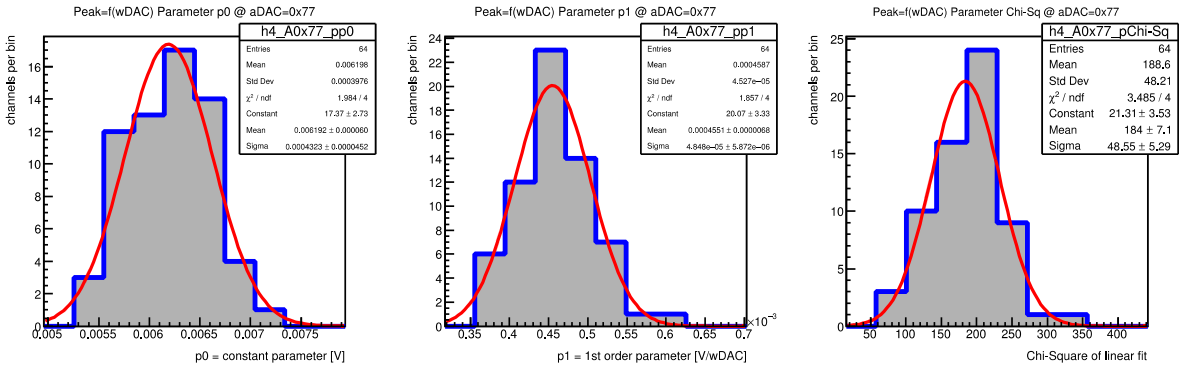


Figura 107 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$x77

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de tensão de pico em função de $wDAC$ (continuação):

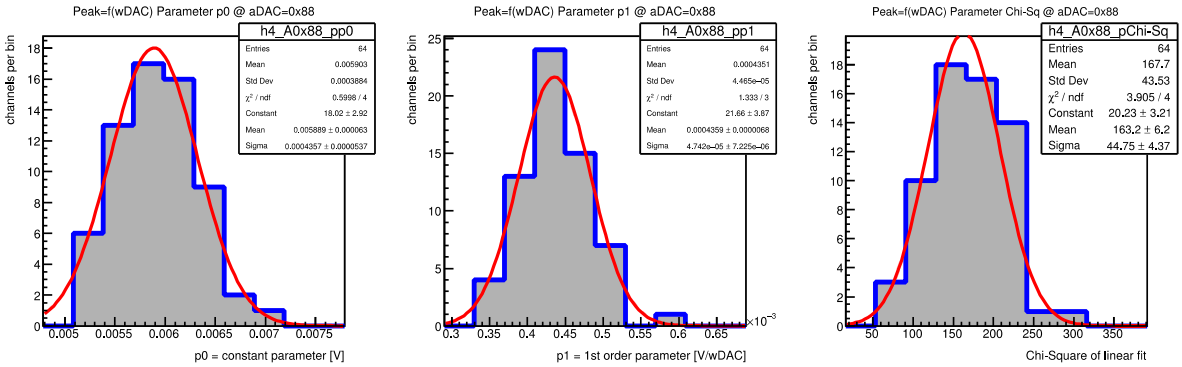


Figura 108 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0x88$

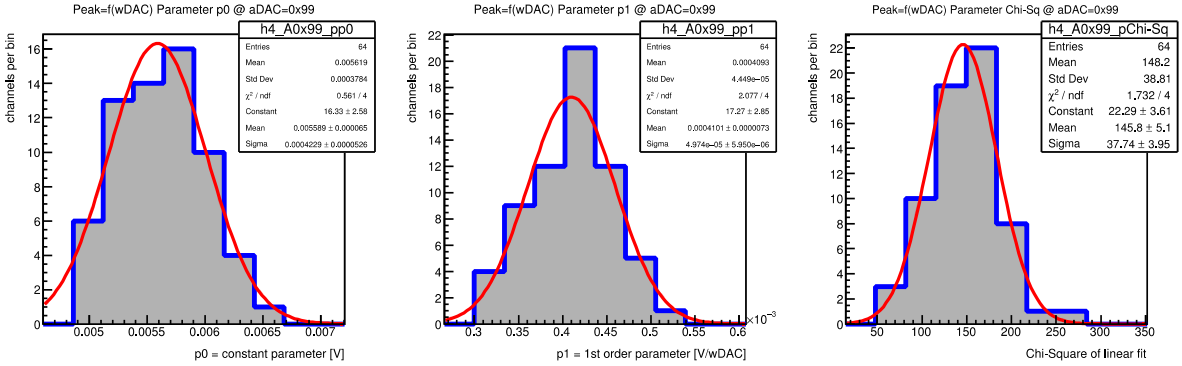


Figura 109 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0x99$

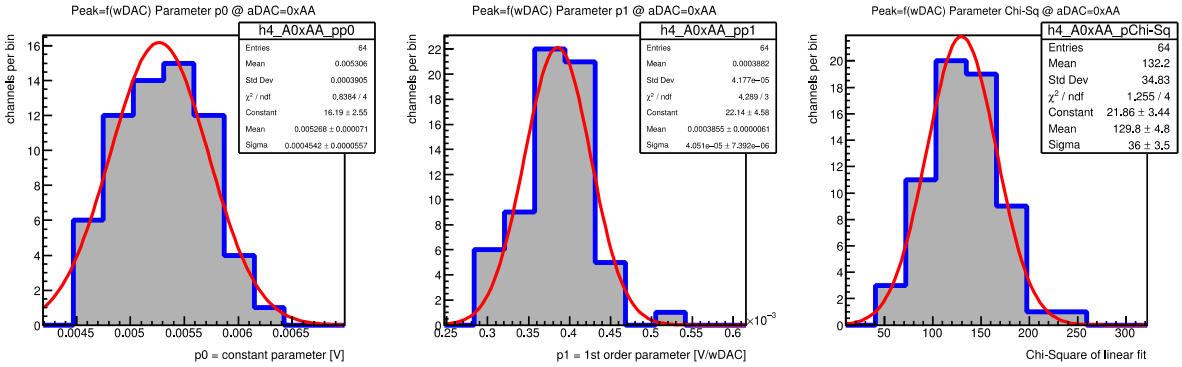


Figura 110 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0xAA$

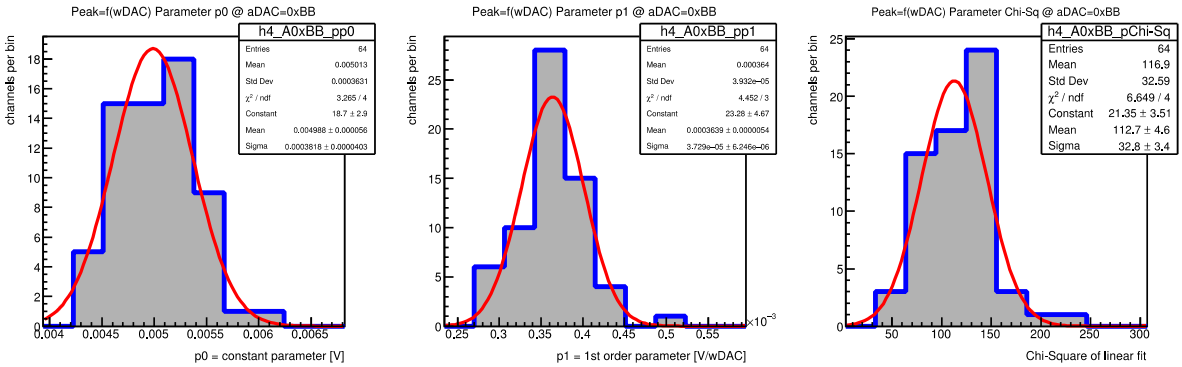


Figura 111 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0xBB$

- Distribuição em várias configurações fixas de $aDAC$ dos parâmetros de ajuste da curva de tensão de pico em função de $wDAC$ (continuação):

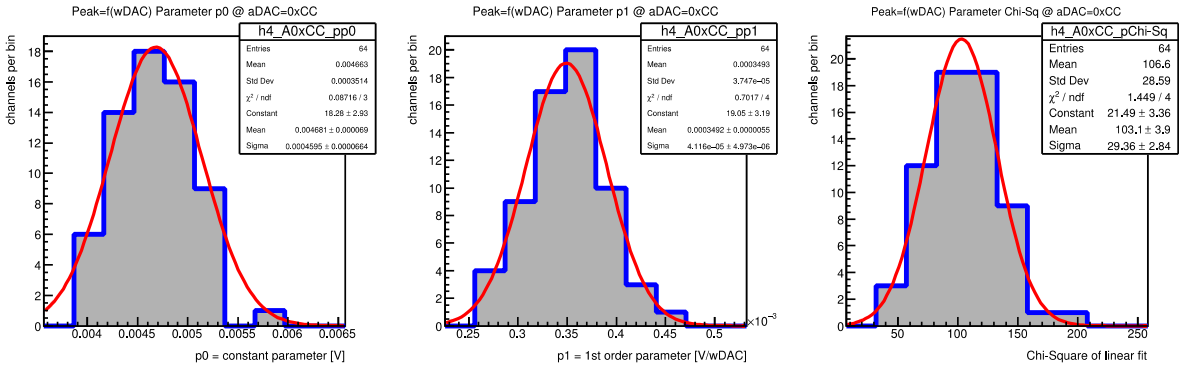


Figura 112 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$xCC

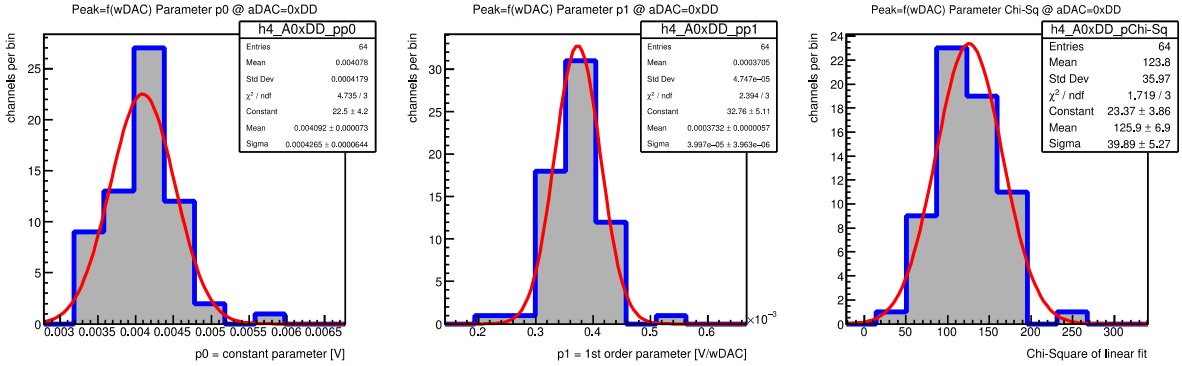


Figura 113 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$xDD

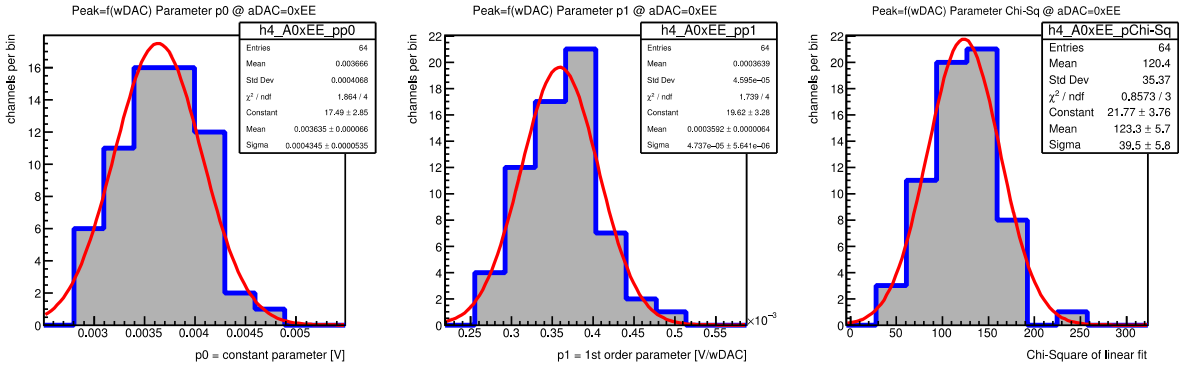


Figura 114 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$xEE

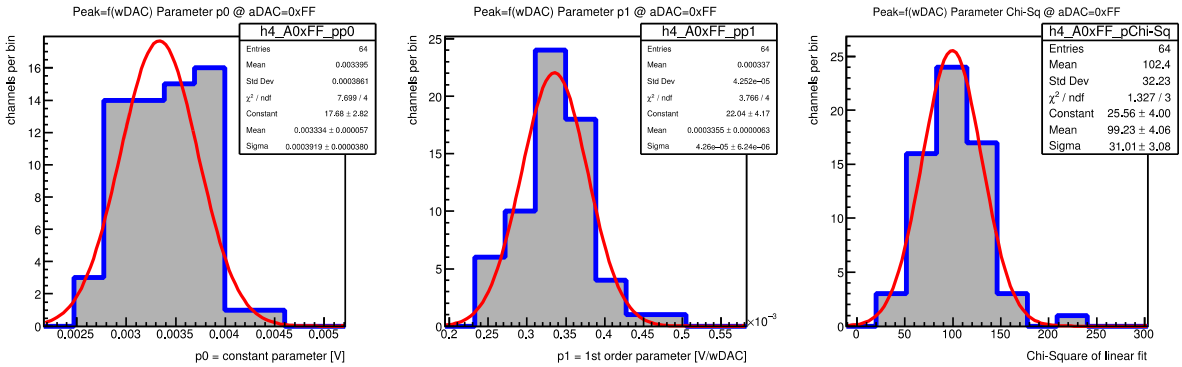


Figura 115 – Parâmetros de v_pico em função de $wDAC$ @ $aDAC = 0$xFF