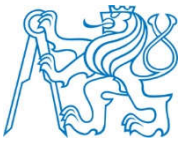


CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATIONS SCIENCES

Bc. Tereza Topková

IDENTIFICATION OF BDS REGISTERS

Master's Thesis

2017

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta dopravní
děkan
Konviktská 20, 110 00 Praha 1

K621..... Ústav letecké dopravy

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení studenta (včetně titulů):

Bc. Tereza Topková

Kód studijního programu a studijní obor studenta:

N 3710 – PL – Provoz a řízení letecké dopravy

Název tématu (česky): **Identifikace BDS registrů**

Název tématu (anglicky): Identification of BDS Registers

Zásady pro vypracování

Při zpracování diplomové práce se řiďte osnovou uvedenou v následujících bodech:

- Přehled a užití standardně užívaných BDS registrů vysílaných v mód S odpovědí
- Vytvoření heuristického algoritmu pro určení typu BDS registru obsaženého ve zprávách DF 20 a 21, jestliže předpokládáme dekódování odpovědí bez znalosti dotazů, jimiž byly vybuzeny
- Otestování algoritmu na vzorku dat naměřených ADS-B přijímačem v rámci laboratoře ATM systémů
- Zhodnocení úspěšnosti vytvořeného algoritmu
- Dekódování meteorologických registrů - vytvoření skriptů pro dekódování vybraných meteorologických informací z přijatých BDS registrů

Rozsah grafických prací: dle pokynů vedoucího diplomové práce

Rozsah průvodní zprávy: minimálně 55 stran textu (včetně obrázků, grafů a tabulek, které jsou součástí průvodní zprávy)

Seznam odborné literatury: ICAO DOC 9871: Technical Provisions for Mode S Services and Extended Squitter

EUROCAE. ED-73E: Minimum Operational Performance Specification for Secondary Surveillance Radar Mode S Transponders

Vedoucí diplomové práce: **Ing. Stanislav Pleninger, Ph.D.**

Datum zadání diplomové práce: **30. června 2016**
(datum prvního zadání této práce, které musí být nejpozději 10 měsíců před datem prvního předpokládaného odevzdání této práce vyplývajícího ze standardní doby studia)

Datum odevzdání diplomové práce: **30. listopadu 2017**
a) datum prvního předpokládaného odevzdání práce vyplývající ze standardní doby studia a z doporučeného časového plánu studia
b) v případě odkladu odevzdání práce následující datum odevzdání práce vyplývající z doporučeného časového plánu studia

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE Fakulta dopravní L. S.

Ing. Jakub Kraus, Ph.D.
vedoucí
Ústavu letecké dopravy

prof. Dr. Ing. Miroslav Svítek, dr. h. c.
děkan fakulty

Potvrzuji převzetí zadání diplomové práce.

Bc. Tereza Topková
jméno a podpis studenta

V Praze dne 30. června 2017

Poděkování

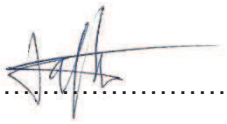
Na tomto místě bych ráda poděkovala všem, kteří mi poskytli podklady a odborné rady pro zpracování této práce. Především děkuji vedoucímu své diplomové práce, Ing. Stanislavu Pleningerovi, Ph. D., za poskytnutí odborných rad a konzultací v celém průběhu tvorby práce. Cením si jeho trpělivosti a vstřícnosti při řešení nestandartních okolností. Poděkování patří také panu Prof. Dr. Falin Wu, který byl mým supervizorem při pobytu na univerzitě Beihang v Pekingu. V neposlední řadě děkuji také ŘLP ČR s.p. za ochotný přístup při poskytnutí potřených dat a odborných znalostí.

Prohlášení

Prohlašuji, že nemám závažný důvod proti užívání tohoto školního díla ve smyslu § 60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

Zároveň prohlašuji, že jsem předloženou práci vypracovala samostatně a že jsem uvedla veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

V Praze dne 30.11.2017



.....

Tereza Topková

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FAKULTA DOPRAVNÍ

IDENTIFIKACE BDS REGISTRŮ

Bc. Tereza Topková

Diplomová práce

2017

Abstrakt

Mód S technologie SSR současně představuje jeden z hlavních prostředků sledování pro ATM. Jedním z typů dotazování SSR módu S vyvolává odpovědi odpovídače módu S na palubě letadla, které obsahují BDS registry. Za předpokladu, že je dotaz radaru neznámý, jsou přijaté odpovědi z odpovídače analyzovány a dále zpracovány s cílem identifikace jednotlivých BDS registrů. K tomuto účelu je vytvořen heuristický algoritmus, který přiřadí číslo BDS registru odpovídající zprávě na základě unikátních bitových řad a jiných kritérií. Správná funkčnost algoritmu je otestována na vzorku dat z ADS-B přijímačů provozovaných Laboratoří ATM systémů na Ústavu letecké dopravy ČVUT v Praze, Fakultě dopravní. Vyhodnocení úspěšnosti algoritmu je provedeno porovnáním ekvivalentních dat z reálného provozu poskytnutých ŘLP ČR, s.p. Meteorologická data z MRAR BDS registrů jsou následně dekódována za účelem získání přesných informací o okolním prostředí z paluby letadla.

Klíčová slova

ATM, SSR, Mód S, BDS register, ELS, EHS, MRAR

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF TRANSPORTATIONS SCIENCES

IDENTIFICATION OF BDS REGISTERS

Bc. Tereza Topková

Master's Thesis

2017

Abstract

The SSR Mode S technology represents currently one of the major means of surveillance in ATM. One type of SSR interrogation may elicit replies from Mode S transponder on board of an aircraft contained BDS register. Under assumption that the radar query is unknown, received Mode S transponder messages are analysed and further processed. A heuristic algorithm is created for identification of a particular BDS register. A BDS register number is assigned to the appropriate message based on the unique bit sequences and another criteria. The correct functionality of the algorithm is tested on the received sample of data from ADS-B receivers operated by the Laboratory of ATM Systems of Department of Air Transport at CTU in Prague, Faculty of Transportation Sciences. The algorithm success rate is evaluated according to the comparison to equivalent data form real operation obtained from the ANS CR. In addition, the meteorological information is decoded from MRAR BDS registers to obtain accurate meteorological data from on-board systems of the aircraft.

Keywords

ATM, SSR, Mode S, BDS register, ELS, EHS, MRAR

Table of Contents

List of Abbreviations 7

Introduction 9

1 Mode S Addressed Surveillance 11

1.1 BDS Registers 12

1.1.1 ELS (Elementary Surveillance) Registers 14

1.1.2 EHS (Enhanced Surveillance) Registers 15

1.1.3 MRAR (Meteorological Routine Air Report) Registers 16

2 Methodology 17

2.1 Data Availability and Collection 18

2.1.1 BDS Registers Availability above Europe 18

2.1.2 Data Receivers 20

2.1.3 Data from ANS CR 21

2.2 Algorithm for Distinguishing Passively Received BDS registers 22

2.3 Algorithm Functionality Check and Success Rate Evaluation 23

2.4 Program for Meteorological Registers Decoding 24

3 Heuristic Algorithm Creation 26

3.1 Initial Data Processing 26

3.1.1 Initial Data Processing of Received Data from ADS-B Receivers 26

3.1.2 Initial Data Processing of ATN CR Data 29

3.2 Data Analysis 30

3.2.1 Bit Analysis of the BDS code $1,0_{16}$ 31

3.2.2 Bit Analysis of the BDS code $1,7_{16}$ 32

3.2.3 Bit Analysis of the BDS code $2,0_{16}$ 32

3.2.4 Bit Analysis of the BDS code $3,0_{16}$ 33

3.2.5 Bit Analysis of the BDS code $4,0_{16}$ 34

3.2.6 Bit Analysis of the BDS code $5,0_{16}$ 34

3.2.7 Bit Analysis of the BDS code $6,0_{16}$ 35

3.2.8 Bit Analysis of the BDS code $4,4_{16}$ 35

3.2.9 Bit Analysis of the BDS code $4,5_{16}$ 36
3.3 Heuristic Algorithm 36
3.3.1 Testing of Unique Parts and Other Specific Criteria 38
3.3.2 BDS Number Assignment According to the Testing 47
3.3.3 Final Saving of Assigned BDS Messages 49
4 Algorithm Functionality Check 51
5 Success Rate Evaluation 54
5.1 ATN CR Data Structure 54
5.2 Algorithm Evaluation 56
6 Meteorological Decoder 60
6.1 Decoding of the BDS register $4,4_{16}$ 60
6.2 Decoding of the BDS register $4,5_{16}$ 65
7 Conclusion 67
References 69
List of Figures 72
List of Tables 73
List of Graphs 74
List of Appendices 75

List of Abbreviations

ACAS	Airborne Collision Avoidance System
ADS-B	Automatic Dependent Surveillance – Broadcast
ANS CR	Air Navigation Services of the Czech Republic
ARA	Active Resolution Advisory
ATC	Air Traffic Control
ATM	Air Traffic Management
ATS	Air Traffic Service
BDS	Comm-B Data Selector
BITE	Built in Test Equipment
CAP	Controller Access Parameters
CRC	Cyclic redundancy check
CTU	Czech Technical University
DAP	Downlink Airborne Parameters
DF	Downlik Format
DME	Distance Measuring Equipment
EHS	Enhanced Surveillance
ELS	Elementary Surveillance
FCU	Flight Control Unit
FOM	Figure of Merit
FTP	File Transfer Protocol
GAT	General Air Traffic
GIBS	Ground-initiated Comm-B
GPS	Global Positioning System
ICAO	International Civil Aviation Organization
IFR	Instrument flight rules
INS	Inertial Navigation Systém
LSB	Least Significant Bit
MB	Message Comm-B
MLAT	Multilateration
MCP	Mode Control Panel
MRAR	Meteorological Routine Air Report
MSB	Most Significant Bit
MTE	Multiple Threat Encounter
MTOW	Maximum Take-Off Weight
PC	Personal Computer
RA	Resolution Advisory
SAC	System Area Code
SAP	System Access Parameters
SDP	Surveillance Data Processing
SIC	System Identification Code
SLM	Standard length message
SSR	Secondary surveillance radar
TAS	True airspeed
TTI	Threat-Type Indicator

UF	Uplink Format
USB	Universal Serial Bus
VNAV	Vercital Navigation
VOR	VHF Omnidirectional Radio Range

Introduction

The Laboratory of ATM Systems at the Department of Air Transport operates ADS-B receivers which may collect any messages transmitted by Mode S transponders located on board of aircraft. The received data is constantly downloaded. It includes several types of messages transmitted on the frequency 1090 MHz. This data may provide direct information from on-board systems. However, it is necessary to process the received messages and invent a way to obtain the information from the received data. One of the replies on a SSR (Secondary Surveillance Radar) interrogation may contain a BDS (Comm-B Data Selector) register, providing Comm-B is considered a 112-bit reply containing the 56-bit MB (Message-Comm B) message field. This field is used by the downlink SLM (Standard Length Message), ground-initiated and broadcast protocols. [1] The BDS register represents the 56-bit MB part of the received message which may contain different information from real operation depending on a BDS register number.

Usually, the SSR Mode S interrogates a particular BDS register number and receives a required reply from the Mode S transponder. Only the replies induced by the SSR Mode S interrogation are received by ADS-B receivers in the laboratory. Thus, the query with the initially required BDS register number is unknown. The transmitted response does not contain any identification which may determine the required number by SSR Mode S query. In order to acquire the information from the replies on the SSR Mode S addressed interrogation, this thesis processes the passively received messages containing BDS registers and it distinguishes particular types of the BDS codes from each other.

This initial data processing may help to the further use of the real operation information from on board of aircraft without any other cooperation. The decoded data may report an aircraft identification, a heading, an altitude, velocities and other flight parameters. This paper is not aimed at decoding the messages with an exception of BDS registers containing a meteorological information. The information on the state of the atmosphere is currently obtained by aerometric probes. This measurement method is mostly done a few times a day and it is time consuming and costly. The aircraft is equipped with several sensors for direct or indirect measurement of environmental parameters which can provide the meteorological data from high altitude as well. Some of the information may be directly decoded in appropriate BDS. There is also possibility to calculate some parameters from different BDS registers, however, the calculation has been proved to be inaccurate. The decoding of meteorological BDS registers is intended for a part of this thesis. [2] [3]

The main purpose of this study is to identify particular BDS register numbers , which are mainly transmitted in Mode S transponder replies to addressed surveillance interrogation

by the SSR. In addition, MRAR (Meteorological Routine Air Reports) BDS registers shall be decoded. The research objectives of this project are summarized below:

- Initial data processing and bit analysis;
- Creation of a heuristic algorithm for distinguishing particular BDS register numbers from each other;
- The algorithm functionality check and evaluation of its success rate; and
- Creation of a program for decoding MRAR registers.

1 Mode S Addressed Surveillance

A secondary surveillance radar (SSR) is one of the means of an independent cooperative surveillance for providing data link communication between ground stations and aircraft, which is further used for air traffic management (ATM). There is an assumption for the system functionality; both subjects are equipped with appropriate Mode S equipment, which contains the SSR Mode S interrogator and the aircraft SSR Mode S transponder. The carrier frequency of the interrogation shall be 1030 MHz and the carrier frequency of the reply transmission shall be 1090 MHz. There are three different type of modes of ground-to-air interrogation in air traffic services (ATS). In order to provide surveillance, the classic SSR may interrogate Mode A to obtain identification information, and Mode C for automatic pressure-altitude transmission. Finally, Mode S SSR interrogation may elicit replies for acquisition of Mode S transponder, broadcast to transmit information to all Mode S transponders, and to provide surveillance and addressed communication with individual Mode S transponder. Mode A interrogation elicits a Mode A reply as well as Mode C interrogation elicits a Mode C reply. However, Mode A/C transponders are not able to reply to Mode S interrogation. In addition, there is intermode: Mode A/C/S all-call (for surveillance of Mode A/C transponders and for acquisition of Mode S transponders) interrogation, which elicits transponder reply according to the query Mode (Mode A/C/S). The last interrogation type is intermode: Mode A/C only all-call (for surveillance of Mode A/C), which obtains appropriate response only from Mode A/C transponders. Mode S transponder do not send a response for the Mode A/C-only all-call. [1] [2]

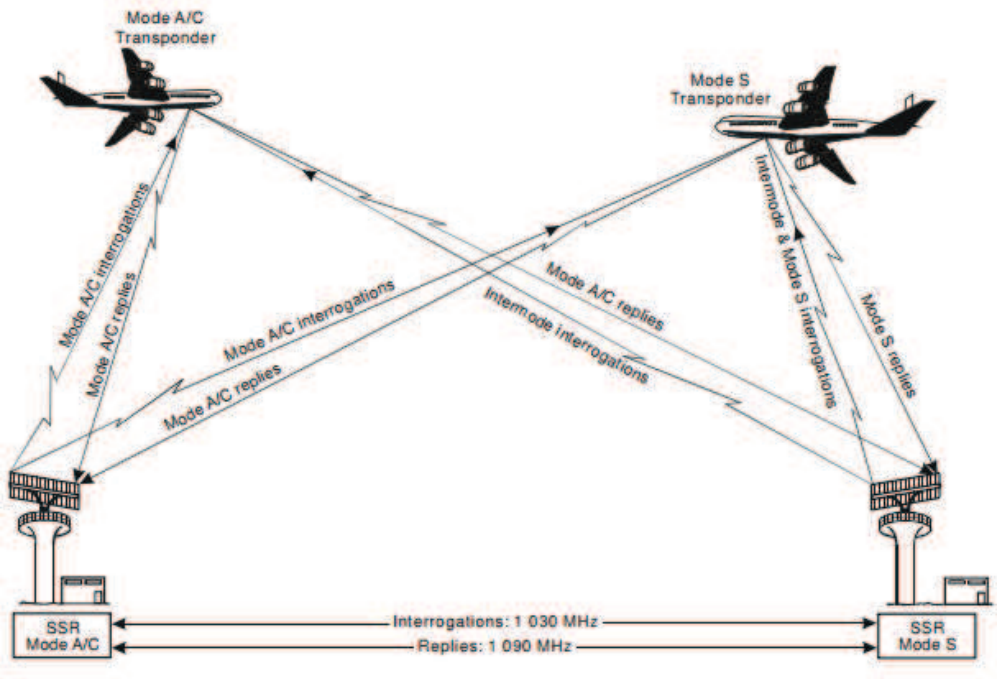


Figure 1. Compatibility between SSR Mode A/C and Mode S [1]

The use of Mode A and Mode C interrogation has proved insufficient for increasing air traffic and number of surveillance devices. SSR Mode S technology provides high-quality information to ensure ATS, and is being used across Europe and other regions with high traffic density. A Mode S addressed surveillance concept is based on the allocation of a unique 24-bit identification code for each transponder, also referred to ICAO 24-bit address. This identifier is allocated in block in accordance with particular states of aircraft registration. An assignment is included in ICAO Annex 10, Volume III [4]. The Mode S technology enables to provide a number of data, which exceeds ATM information and is suitable for implementation into air traffic services. Depending on aircraft equipment and ground station interrogation, a Mode S reply can contain aircraft's identity, its altitude, velocity, heading and meteorological information from on-board systems etc. These replies are protected by a robust error detection/correction scheme which ensures high reliability of the transferred information. Introduction of Mode S technology increases the air surveillance quality, overall safety, and makes providing ATS more efficient. [2] [4]

1.1 BDS Registers

There shall be two essential fields contained in every Mode S transmission. The beginning of the Mode S interrogation is designed by 5-bit of the uplink format (UF), whereas a format of all Mode S replies are defined by 5-bit of the downlink format (DF) in the beginning of the message. The second essential field consists of 24-bit of parity information overlaid either on the aircraft address (AP) or on the interrogator identifier (PI) at the end of the transmission. [2]

In response to Mode S interrogation there are four Mode S replies categories transmitted:

- Mode S all-call replies (DF 11);
- Surveillance and standard-length communications replies (DF 4, 5, 20 and 21);
- Extended length communication replies (DF 24); and
- Air-air surveillance replies (DF 0 and 16). [2]

Two basic types of radar query and transponder reply are used for addressed surveillance and standard length communication transactions. The first type represents a short transmission, and is characterised by UF 4 and 5, and DF 4 and 5. The second type is long interrogation and reply, it is contained in messages with descriptors UF 20 and 21, and DF 20 and 21. For the purposes of this research, only Mode S transponder replies DF 20 and 21 will be discussed in further details. [2]

Mode S transponder replies with DF 20 and 21 are elicited by addressed radar queries and are contained in 112-bit data field. In Figure 2, there is an examplary illustration of bit allocation in DF 20 of Comm-B altitude reply.

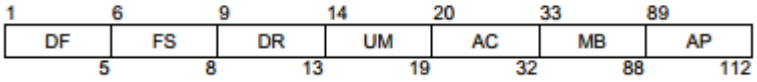


Figure 2. Comm-B altitude reply, DF 20

DF – Downlink Format

FS – Flight Status

DR – Downlink Request

UM – Utility Message

AC – Altitude Code

MB – Message, Comm-B

AP – Address/Parity. [2]

Required flight data can be provided to the ground station through the GIBS (Ground-initiated Comm-B) protocol. The GIBS protocol elicits a Comm-B message from a Mode S transponder on board the aircraft. This transponder may obtain a 56-bit buffer with real-time information from aircraft avionics, such as aircraft state data within 255 transponder registers. These 56-bit sequences are known as BDS (Comm-B data selector) registers such as GIBS[20]. Real-time data shall be coded in binary code, and shall contain up-to-date information depending on the number of BDS register. If the contained information has not been updated within the time allowed, the register shall be deleted. [1] There is used for BDS registers a designation with the following notation: X, Y_{16} in this thesis. The values are number 0-F of hexadecimal code, which is characterised by lower index number.

According to the Mode S implementation in European Regulation, "*all aircraft operating IFR/GAT in Europe are required to carry and operate Mode S Level transponder(s) with Mode S Elementary Surveillance (ELS) capability. All State aircraft operating IFR/GAT in Europe are required to have aircraft with the Mode S ELS equipment by the 7th December 2017. All aircraft operating IFR/GAT in Europe, MTOW exceeding 5700 kg or a maximum cruising TAS capability greater than 250 knots are required to be equipped with and operate Mode S transponder with Elementary Surveillance (ELS) and Enhanced Surveillance (EHS), (as well as ADS-B 1090 MHz extended squitter). This regulation is applicable for aircraft with a certificate of airworthiness first issued on or after 8th June 2016, and the rest of the aircraft shall be retrofitted by 7th June 2020.*" [7] Besides BDS registers involved in ELS and EHS, there is a concentration on MRAR (Meteorological routine air report) for obtaining more

accurate meteorological data. According to ICAO Doc 9871, Technical Provisions for Mode S Services and Extended Squitter, there are contained data with its bit position in Appendix A. [5]

1.1.1 ELS (Elementary Surveillance) Registers

Mode S Elementary Surveillance shall provide transponder parameters such as the ICAO 24-bit aircraft address, SSR Mode 3/A, altitude reporting in 25ft increments and a flight status (airborne/on ground). In addition, BDS registers $1,0_{16}$; $1,7_{16}$; $2,0_{16}$ and $3,0_{16}$ are required to be transmitted within ELS. Content of ELS registers is specified in Table 1 below. [1]

No.	BDS register	BDS register Name
1	$1,0_{16}$	Data link capability report
2	$1,7_{16}$	Common usage GIBC capability report
3	$2,0_{16}$	Aircraft identification
4	$3,0_{16}$	ACAS active resolution advisory

Table 1. ELS BDS registers [3]

The BDS register $1,0_{16}$ contains essential information about configuration of Mode S transponder and operational status of datalink. In order to find available data provided by individual Mode S transponder, the BDS register $1,7_{16}$ shall be sent. The binary sequence of the 'Common usage GIBC capability report' designates other status of BDS registers with dynamic data. On the assumption that correct real-data is available and regularly updated, the relevant bit of the BDS code $1,7_{16}$ is set to '1'. However, the bit value '1' means that any of the group of the BDS information may be ensured. For example, when at least one BDS $5,0_{16}$ data block (from roll angle, true track angle, ground speed, track angle rate, true airspeed) is updated correctly, the appropriate bit in the BDS code $1,7_{16}$ will be set to '1', although data from any other block is not provided. Information from the BDS code $2,0_{16}$, the 'Aircraft identification', correlates with flight plan data and with surveillance data processing (SDP) systems. The last BDS register of ELS ($3,0_{16}$) enables to extract real-time ACAS system status from the cockpit to the ground stations. The 'ACAS Resolution Advisory' shall be transmitted on condition that the ACAS system has reported RA. This BDS code is not typically interrogate by SSR. Nevertheless, it is sent by Mode S transponder from the airplane and can be received with DF 20 or 21. The RA may be displayed on the air traffic controller screen. There is discrepancy about implementing ACAS active resolution advisory to the ATS. Currently, while there is RA notification from an aircraft pilot, an air traffic controller is not responsible to ensure separation minima for the aircraft involved. If BDS code $3,0_{16}$ is implemented, the air traffic controller receives a RA notification on screen when this occurs. However, the pilot does not. Then, stressful and hazardous situations may be avoided if air traffic control is not notified. [3] [6]

1.1.2 EHS (Enhanced Surveillance) Registers

Every Mode S Enhanced Surveillance transponder shall provide all services of ELS automatically. EHS is extended by next parameters which are known as DAPs (Downlink Airborne Parameters), it ensures additional data for air traffic controller (CAPs – Controller Access Parameters), and for upgrade of ATM systems functionality (SAPs – System Access Parameters). GIBS protocol shall enable transmission of BDS registers, which are specified in Table 2. [1]

No.	BDS register	BDS register Name
1	$4,0_{16}$	Selected vertical intention
2	$5,0_{16}$	Track and turn report
3	$6,0_{16}$	Heading and speed report

Table 2. EHS BDS registers [3]

The main objective of implementing EHS BDS register is to improve ATC systems of current and real-time information in order to predict future trajectory of the flight. Selected altitude information contained in BDS register $4,0_{16}$ may be obtained from various on-board inputs, depending on aircraft configuration and individual aircraft avionics. Provided data is limited to vertical intention of the aircraft, which is information processed in surveillance devices and used for ATS in particular. Other EHS registers shall transmit the 'Track and turn report' (BDS register $5,0_{16}$) and the 'Heading and speed report' (BDS register $6,0_{16}$). These registers are characterised by similar bit structure with five different parameters, each complementing extracted flight data. Particular parameters are further specified in Table 3 below. The given time for data update, inside these data sets, is set to 1 second. If the data update does not occur within 2 seconds, the status bit for following parameters, as well as the bits of the parameter, shall be set to '0'. [6]

BDS register	Data Set
$4,0_{16}$	Selected altitude
$5,0_{16}$	Roll angle
	True track angle
	Ground speed
	Track angle rate
	True airspeed
$6,0_{16}$	Magnetic heading
	Indicated Airspeed
	Mach number
	Barometric altitude rate
	Inertial vertical velocity

Table 3. EHS BDS registers provided data sets [5]

1.1.3 MRAR (Meteorological Routine Air Report) Registers

Meteorological Routine Air Report registers are neither part of ELS, nor of EHS. They may however provide meteorological data obtained directly from sensors, which aircraft may be equipped with. In addition, the on-board system may process meteorological data. Due to this fact, there is no need for further data processing in a ground station. Two BDS registers $4{,}4_{16}$ and $4{,}5_{16}$ may be transmitted in Mode S replies, see below in Table 4. The MRAR data availability is detectable from bits on positions 13 and 14 setting in the BDS register $1{,}7_{16}$. On condition that at least one of these bits is set to '1', the meteorological data may be extracted from Mode S transponder of the aircraft. [3] [6]

No.	BDS register	BDS register Name
1	$4{,}4_{16}$	Meteorological routine air report
2	$4{,}5_{16}$	Meteorological hazard report

Table 4. MRAR BDS registers [3]

Standard meteorological information is transmitted in the BDS register $4{,}4_{16}$ and hazardous meteorological information in the BDS register $4{,}5_{16}$. Some of the contained data sets are duplicates, as specified in Table 5. The given time for data update inside data of 'Meteorological Routine Air Report' sets is set to one second, as for the BDS code $5{,}0_{16}$ and $6{,}0_{16}$. [6]

BDS register	Data set
$4{,}4_{16}$	Wind speed
	Wind direction
	Static air temperature
	Average static pressure
	Turbulence
	Humidity
$4{,}5_{16}$	Turbulence
	Wind shear
	Microburst
	Icing
	Wake vortex
	Static air temperature
	Average static pressure
	Radio height

Table 5. MRAR BDS registers provided data sets [5]

Based on the contained parameters in BDS registers $5{,}0_{16}$ and $6{,}0_{16}$, meteorological information may be obtained by indirect calculation. The resulting data does not ensure high level of accuracy. [8]

2 Methodology

In this research there Mode S transmitted replies received by two different resources are analysed; data from Radarcape receivers and data provided by ANS of the Czech Republic. Both data samples are filtrated and further processed by created heuristic algorithm. The algorithm is based on data analysis. In addition, the algorithm is improved by comparison its results of BDS number assignment with known BDS number from ATN data sample. The algorithm functionality is tested on data from the Laboratory of ATM Systems, which ensures data for next processing. The final comparison of BDS register assigned by the algorithm with already allocated BDS numbers of ATN data enables to evaluate success rate of the algorithm. Furthermore, meteorological registers from the data sample are decoded by created meteorological decoder. Finally, all of the obtained results are presented in conclusion. The process itself is characterised in Figure 3 below.

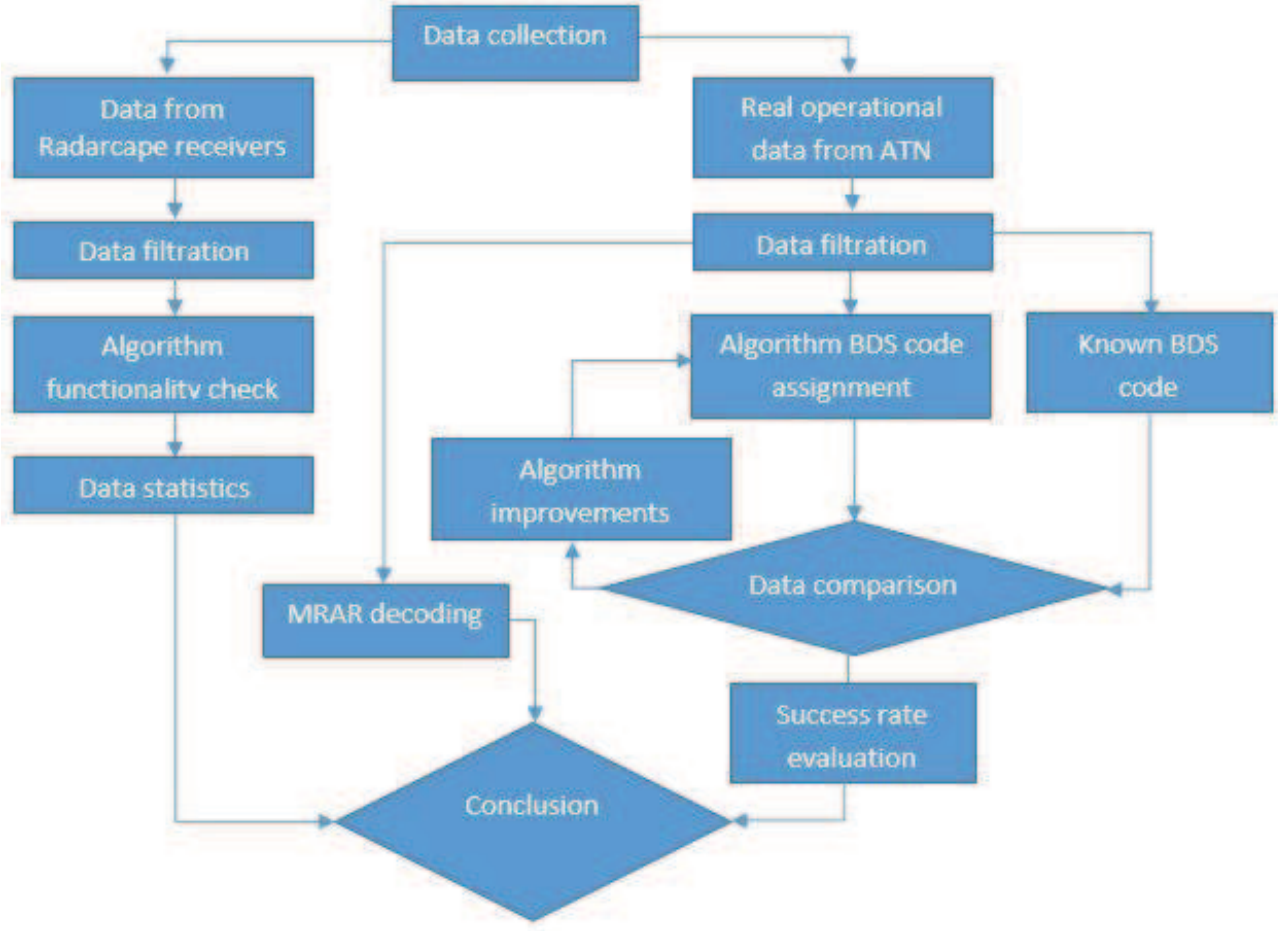


Figure 3. Thesis workflow chart

2.1 Data Availability and Collection

Two conditions need to be met for ensuring data availability for this research:

- There are BDS registers of elementary and enhanced surveillance, as well as MRAR BDS registers transmitted in Mode S reply; and of course,
- There is a receiver enabled to receive them.

2.1.1 BDS Registers Availability above Europe

As aforementioned, the ELS Mode S transponder is mandatory for IFR/GAT aircraft and MTOW exceeding 5700 kg or a maximum cruising TAS capability greater than 250 knots, along with an EHS Mode S transponder. [7] There are currently no European regulations for requiring MRAR equipment. Therefore, there is only a small number of SSR are implemented with MRAR technology. In addition, only a small proportion of aircraft (around 2-3%) is equipped with Mode S transponder provided MRAR registers. [3]

Almost the entire European region is covered by SSR system, which enables Mode S ELS to interrogate. Mode-S EHS is based on the same principle as ELS, and the number of SSR EHS interrogation has enormous potential to be upgraded from ELS in the future. However, several difficulties must be overcome when implementing of EHS. Increasing number of aircraft may provide data and an advantage of improvement quality of ATS. Finally, for Mode S transmission of meteorological information, the SSR needs to be extended for interrogating additional MRAR registers. The MRAR BDS register $4{,}4_{16}$ is currently interrogated by every second radar rotation above Czech Republic. Unfortunately, there is no interrogation of BDS code $4{,}5_{16}$ due to the answer rarely containing any information. The capacity of SSR queries is used for more important operational registers. [3] [8]

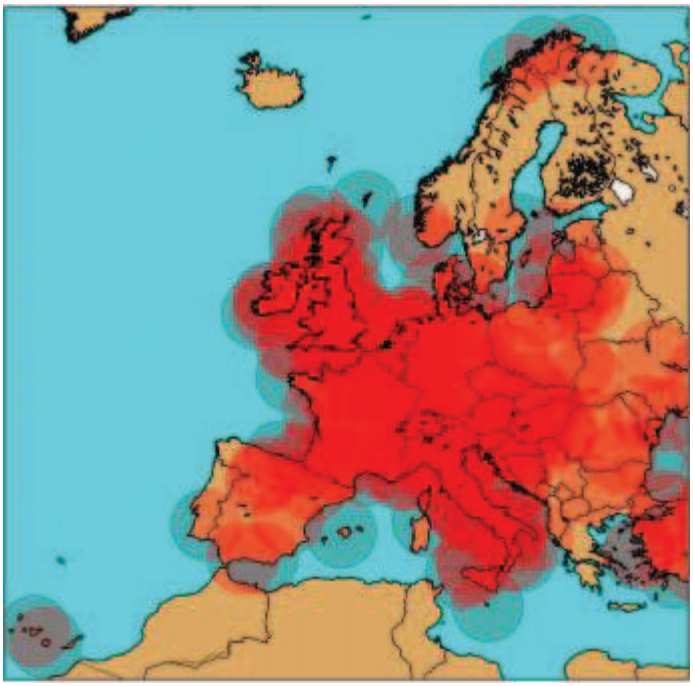


Figure 4. SSR Mode S ELS coverage above Europe [8]

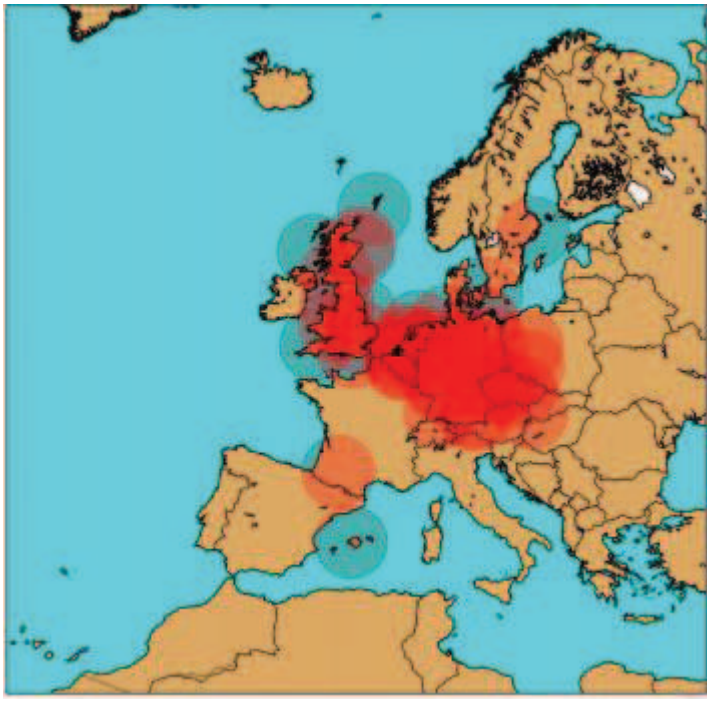


Figure 5. SSR Mode S EHS coverage above Europe [8]

Figure 6. SSR Mode S MRAR coverage above Europe [8]

According to Figures 4-6 above, the SSR interrogation of ELS, EHS and MRAR registers is ensured within the region of Czech Republic. The Air Navigation Services of Czech Republic (ANS CR) currently operates three SSR enabled services, which are further specified in Table 6. [3] [8]

No.	SSR Name	SAC*	SIC**	Interrogation	Latitude	Longitude	Range [NM]	Height [m]
1	PRAHA	49	135	ELS, EHS, MRAR $4,4_{16}$	50 05 11.91 N	14 16 13.15 E	160	383
2	BUKOP	49	144	ELS, EHS, MRAR $4,4_{16}$	49 39 34.71 N	16 08 00.22 E	200	845
3	PISEK	49	130	ELS, EHS, MRAR $4,4_{16}$	49 47 05.52 N	14 02 04.36 E	160	622

Table 6. SSR operated in the Czech Republic

***SAC-System Identification Code**

****SIC-System Area Code [3]**

While the coverage of SSR transmissions is considered sufficient, some elicited replies may be caused by SSR from surrounding areas (e.g. JAVOR, AUERSBG or VIENNA SSR). Nevertheless, these SSR are able to provide only ELS and EHS questioning. [3]

2.1.2 Data Receivers

The Laboratory of ATM Systems at Department of Air Transport (CTU in Prague, Faculty of Transportations Sciences) gradually acquires four ADS-B receivers. The last of them was implemented in early 2017. The Radarcape receivers do not need any additional software as they are able to work by themselves without any PC support. Web and USB interface is

used for data collection. The receivers are equipped with a passive antenna with the relevant frequency range 1090 MHz for receiving Mode S transponder replies. The technology is completed by a built-in GPS receiver, that provides another possibility to use this system for MLAT (Multilateral) navigation. [9]

The Ethernet interface enables output data format such as a binary or hexadecimal format, a time assignment (GPS or system time), DF of received messages, CRC (Cyclic redundancy check) or to receive Mode A/C. Although, the receivers were characterised as ADS-B receivers, they can receive any information transmitted on frequency 1090MHz as well. [9]

Data is received from all receivers, depending on the condition of particular receivers, and is continuously extracted. This real-time, or stored data, can be downloaded via FTP (File Transfer Protocol) server. Data is automatically recorded and saved into a ‚.dat' format which is named according to the date, hours and minutes of receiving. The Figure 7 demonstrates a sample of the short as well as the long received messages.

```
1480639200064;1;025843DD0A17;5D471F898D2494
1480639200114;1;02584544FDBC;A0001690205894B3C302E0FEE9DE
1480639200114;1;025845BE6E0D;8D471F899941E1922804921C557D
1480639200114;1;0258464A2D62;079B5EDF92620F
1480639200165;1;0258493B5212;84A5ACBEC5BEA835EAFB1838FE91
1480639200170;2;02584AB0CE7F;A0001718EDEA0B313FF40082669A
1480639200216;1;02584AB0D3EE;A0001718EDEA0B313FF40082669A
1480639200216;1;02584AF35021;A0001718EDEA0B313DF40082669A
1480639200216;1;02584B30646C;A8000A1CC6500030A800002513B9
1480639200216;1;02584B727C48;A0001718EDEA0B313FF40082669A
1480639200216;1;02584C707D2F;A8000A1CC6500030A800002513B9
1480639200216;1;02584D5F53F2;200017B00BA6FE
```

Figure 7. Sample of data received by Radarcape receivers

2.1.3 Data from ANS CR

Samples of data from real operation are provided by Air Navigation Services of the Czech Republic. This data is obtained on the basis of the SSR interrogation and ASTERIX data is received and further processed to ensure information from ELS, EHS, as well as from MRAR BDS registers. The ANS CR uses a program of Croatia Control with decoding software for ASTERIX data processing which is online available. [11] Data is provided in two formats; ‚.pcap' and ‚.xml'. The ‚.pcap' format represents original source data and the ‚.xml' format illustrates decoded data by Croatian program, which are specified in Figure 8.

```
<ASTERIX ver="1" cat="48">
  <SAC>49</SAC>
  <SIC>130</SIC>
  <ToD>59104.0937500</ToD>
  <RHO>98.3906250</RHO>
  <THETA>81.5405273</THETA>
  <FL>360.2500000</FL>
  <ACAddr>406532</ACAddr>
  <TId>BAW887</TId>
  <MBdata>EA5A0B31600C01</MBdata>
  <DSB1>6</DSB1>
  <DSB2>0</DSB2>
  <MBdata>C6500030900000</MBdata>
  <DSB1>4</DSB1>
  <DSB2>0</DSB2>
  <STAT>0</STAT>
</ASTERIX>
```

Figure 8. Sample of ATN data

SAC – System area code

SIC – System identification code

RHO, THETA – Position information

FL – Flight level

ACAddr – ICAO address

TId – Flight identification

MBdata – BDS register data

DSB1 – BDS1 code

DSB2 – BDS2 code

The provided data contains the known BDS register number and the corresponding message in hexadecimal scale. Therefore, this data is suitable for testing by the created algorithm and the result of assigned number can be used for evaluating its success rate.

2.2 Algorithm for Distinguishing Passively Received BDS registers

A heuristic algorithm is created for the purpose of identifying particular BDS registers in case of passive receiving the reply from Mode S transponder elicited by the SSR Mode S interrogation. An expected result of the algorithm shall be distinguishing the BDS register (mentioned above) from each other. In other words; every received 56-bit code shall be assigned to one specific number of BDS register. Nevertheless, decoding received message is not the main task of the algorithm. The particular decoder is created for decoding BDS register $4{,}4_{16}$ and $4{,}5_{16}$ (see the Chapter 2.4).

After data filtration, there shall be only 56-bit binary arrays representing contents of BDS registers from received messages, with DF 20 or DF 21. Initially, a bit analysis must

be processed to find unique combinations of binary code, which cannot be contained in any other register. These significant parts are found in every BDS, on the base of its bit content according to Appendix A. For example, there must be coded BDS1 code 2 (the first four bits) and BDS2 code 0 (the bit on positions 5-8) for the BDS register $2,0_{16}$ and bits on positions 25, 33 and 35 must be set to '0'. Then, if a status bit is set to '0', the following bits with required information must be also set to '0'. Thus, the position and setting of the bit defines the register on the sequence of the BDS received message. [6]

Due to many bit combinations and uncertain bit assignments, any type of deterministic algorithm was excluded from the solution. The heuristic algorithm may apply iteration methods with progressive adding of processing groups of bits, which can gradually define a BDS register. An integral part of this method is considered a loop with conditional computer programming (if-then(-else), else-if construction). A sequence of bits is tested and it determines which registers may contain these bits. The testing is extended by different criteria. There is check of status bit and its corresponding field and the fields are decoded and compared to possible value, which can be achieved in particular case.

Considering all bit combinations and programming methods, the algorithm shall be programmed in numerical computing environment of MATLAB.

2.3 Algorithm Functionality Check and Success Rate Evaluation

MATLAB program automatically analyses the feasibility of a script or a function and warns the user of errors. The issue following the algorithm creation is whether there is only one BDS assignment; and whether this assignment is correct. There are two methods to test the algorithm:

- Functionality check on received data provided by the Laboratory of ATM Systems;
- Comparison BDS number with BDS registers provided by ANS CR to evaluate the algorithm success rate

Firstly, the decoded data from ADS-B messages is used for the functionality check. A Mode S receiver of the laboratory can provide all messages sent in Mode S reply on frequency 1090 MHz. Therefore, the data filtration does not require to choose only replies of the SSR Mode S interrogation and save them into applicable format for the algorithm. Then message after message is tested and each of them is assigned one BDS number.

Secondly, the known number of BDS register ensures the possibility to evaluate the assignment of algorithm BDS number success. For this purpose, the obtained data from ANS CR has to be filtered. The specific information about BDS number and the corresponding

content of 56-bit message has to be chosen from the list of information contained in provided data formats. The message has to be converted from hexadecimal to binary scale to unify input from passive receivers in the laboratory with the data from ANS CR. Thus, both samples of data can be tested by one version of the heuristic algorithm without any obstacles. The comparison of provided BDS number with assigned BDS number shows, if the algorithm is able to identify the numbers of BDS registers correctly. Finally, the successful rate of the algorithm is evaluated calculated for the whole algorithm as well as for particular BDS registers.

2.4 Program for Meteorological Registers Decoding

The next part focuses on decoding meteorological information from MRAR BDS registers. Particular binary fields of the 56 bits will be converted to provide appropriate information. The Least Significant Bit (LSB) and the Most Significant Bit (MSB), together with a range of values, build the base for obtaining required data. The MSB is situated on the left side of the binary field and represents the greatest numerical value. In addition, the LSB is situated on the right side of the binary field and represents the least numerical value. A relevant value shall be assigned for each bit combination, the binary code will be converted to decimal system, and a value will be calculated on the base of data set scale. The overview of meteorological information which may be obtained from the BDS code $4,4_{16}$ is characterized in Table 7 below.

Data field	Bit position	Number of bits	Range	LSB	MSB	Unit
FOM/SOURCE*	1-4	4	1 - 15	-	-	-
Wind speed	6-14	9	0 - 511	1	256	[kn]
Wind direction	15-23	8	0 - 360	180/256	180	[°]
Static air temperature	25-34	10	-128 - +128	0.25	64	[°C]
Average static pressure	36-46	11	0 - 2047	1	1024	[hPa]
Turbulence*	48-49	2	-	-	-	-
Humidity	51-56	7	0 - 127	100/64	100	[%]

Table 7. Data fields specifications for BDS $4,4_{16}$

*** Fields have word expression [5]**

Besides routine meteorological information, the BDS code $4,4_{16}$ contains bits with specified data resource (the so called FOM/SOURCE field), which cannot be expressed with numerical value. The names of the resource will be assigned by their shortcut expression. Similarly, the filed containing information about turbulence will be expressed by qualified strength of the phenomena.

Some of the contained data in the BDS register $4,5_{16}$ are duplicates of routine meteorological data, which ensure providing complete meteorological information, in case only hazard report is received. Meteorological hazard information (turbulence, wind shear, microburst, icing, wake vortex) is characterised by two bits assigned information. The whole content of the BDS code $4,5_{16}$ is described in Table 8.

Data field	Bit position	Number of bits	Range	LSB	MSB	Units
Turbulence	2-3	2	-	-	-	-
Wind shear	5-6	2	-	-	-	-
Microburst	8-9	2	-	-	-	-
Icing	11-12	2	-	-	-	-
Wake vortex	14-15	2	-	-	-	-
Static air temperature	18-26	10	-128 - +128	0,25	64	[°C]
Average static pressure	28-38	11	0 - 2047	1	1024	[hPa]
Radio height	40-51	12	0 - 65528	16	32768	[ft]
Reserved	52-56	5	-	-	-	-

Table 8. BDS register $4,5_{16}$ decoding information [5]

In order to decode meteorological information from relevant registers, the MATLAB decoding function and script will be created according to the first edition of Doc 9871 [5] containing a current version of bit structure in the meteorological codes.

3 Heuristic Algorithm Creation

Before the heuristic algorithm creation, the initial data processing has to be ensured to obtain data samples in a required form. The conditions for distinguishing particular BDS numbers are specified in the bit analysis of each BDS register. Afterwards, the algorithm may be created to assigned a corresponding BDS number to a messages received as a reply to the Mode S addressed interrogation.

3.1 Initial Data Processing

For the purposes of this project, data are ensured from two resources; ADS-B receivers operated by the Laboratory of ATM systems and the ATN data from real operation. Firstly, the Mode S transponder responses on the SSR interrogation are continuously received on a frequency 1090 MHz by a passive antenna and initially processed by the Radarcape receiver, as well as all messages transmitted by the transponder. The provided data from varying time periods may be downloaded through using FTP server. A MATLAB script 'Data_initial_processing_ADSB.m' has been created for initial data processing. Secondly, the data provided by ATN also shall be initially processed to obtain a required format of the data. An another MATLAB script 'Data_initial_processing_ATN.m' has been created for the filtration of the data from real operation. The both scripts have been compiled only for needs of this work. Another part for a selection of required messages and an exclusion of an incorrectly received data has been added. Two important steps are included in the programs; the data collection and filtration of BDS messages. The programs are specified in Appendix B and Appendix C.

In short time, there is a large number of received Mode S replies, therefore the data processing can take several minutes but also hours depending on the size of a data sample. Actually, the thesis is aimed at a creation of the heuristic algorithm and the meteorological decoder and initial data processing scripts have not been optimized since obtaining the required data.

3.1.1 Initial Data Processing of Received Data from ADS-B Receivers

The received data is loaded from '.dat' format with help of a MATLAB import function. A script has been created from the already existing program which is used for initial data processing for further applications of the received data in the MATLAB interface within the laboratory by default. The collected data may obtain the exact time of its receiving, the receiver address, as well as content of received messages. Defined variables are padded with the data from received messages in a decimal, hexadecimal or binary code, see Figure 9.

```
%Defining variables of initial message processing
        DF=NaN(mm,1); %Downlink format
        CRC=cell(mm,1); %Cyclic redundancy check
%         Adress24=cell(mm,1);% ICAO address hexadecimal
        MessageBIN=cell(mm,1);% Message in binary code
        BDS=cell(mm,1); %BDS message in binary code
        BDSmessages=[];
%         Time_s=NaN(mm,1); %time saving in s
%         Time_ns=NaN(mm,1); %time saving in ns
```

Figure 9. Defined variables for initial data processing

Converting among particular scales is ensured by standard MATLAB functions and supplementary functions 'bin2hex.m' and 'hex2bin2.m'. In order to obtain the address of an aircraft, another function 'crc_division4.m' may be used in this script. A particular message is tested in a for-loop to assign its type. In order to find messages containing BDS registers, only binary sequences with length of twenty-eight hexadecimal bits are chosen. This ensures that the length of the message is equivalent to the long reply of the addressed surveillance. A double numeric data type of the DF component provides information about the downlink format of the received message. In other words, this number distinguishes particular categories of Mode S replies. The downlink format is decoded and in case that it is equal to '20' or '21', variables are filled with the corresponding data. The BDS register in a binary code is situated between bits on positions 33 to 88 of the original message. If conditions of the length and the value of the downlink format are met, the appropriate row of variable 'MessageBIN' is filled with 56-bits of the BDS code. A cell with a CRC array is filled with a sequence of six characters (e.g. '9FF20C'). A variable 'Adress24' with hexadecimal the 24-bits ICAO address and variables containing time in seconds and nanoseconds are not needed and marked as comment. The data filtration for messages with the DF 20 and the DF 21 is specified in Figure 10.

```
if delka==28
        zpravaBIN=hex2bin2(Message{k,1}); % tested message
          timeBIN=hex2bin2(time{k,1});
          time_s=polyval(timeBIN(1,1:18),2);%binary to decimal transformation
          time_ns=polyval(timeBIN(1,19:48),2);%nanoseconds

        if zpravaBIN(1,1:5)==[1 0 1 0 0] %if DF20
            DF(k,1)=20;
            BDS{k,1}=zpravaBIN(1,33:88);
            [Adress,Adress_hex]=crc_division4(zpravaBIN);
            CRC{k,1}=Adress_hex;
              Adress24{k,1}=Adress_hex;
            MessageBIN{k,1}=zpravaBIN;
              Time_s(k,1)=time_s;
              Time_ns(k,1)=time_ns;

        elseif zpravaBIN(1,1:5)==[1 0 1 0 1] %if DF21
            DF(k,1)=21;
            BDS{k,1}=zpravaBIN(1,33:88);
            [Adress,Adress_hex]=crc_division4(zpravaBIN);
            CRC{k,1}=Adress_hex;
              Adress24{k,1}=Adress_hex;
            MessageBIN{k,1}=zpravaBIN;
              Time_s(k,1)=time_s;
              Time_ns(k,1)=time_ns;
        end
```

Figure 10. Filtration of long messages with DF=20 or DF=21

The collected data are further checked whether it is correctly received or not. A CRC (Cyclic Redundancy Check) cell provides information for excluding errors by data receiving. Real-time data errors may occur while more than one message is received in the same time and the system decodes them faulty. In case, the transponder Mode S replies are received correctly, the CRC cell shall contain the 24-bit aircraft address. The length of CRC is saved into a 'CRC_check' variable. There may be blank cells in the CRC array. For that reason, length of CRC fields is tested and only values with six characters are sorted out. Erroneous values are further removed, when the address is decoded as '000000'. If all conditions are met, the 56-bits of binary sequence is saved into 'BDSmessage' variable, which is in the double format. The described testing is characterized in Figure 11 below.

```
CRC_check=(length(CRC{k,1})); %for length check
if CRC_check==6 %CRC length check
   Zero_CRC=strcmp(CRC{k,1},'000000'); %erroneous CRC exclusion
   if Zero_CRC==0;
   BDSmessages=[BDSmessages;BDS{k,1}]; %filtrated BDS message saving
   end
end
```

Figure 11. CRC and bit setting check

An output of initial data processing with the ‘BDSmessages’ variable is saved into a MAT-file with the original name of the input file. Variables contained the filtrated data are saved into a ‘Test’ folder and are prepared for using by the heuristic algorithm.

3.1.2 Initial Data Processing of ATN CR Data

The real operational data from ATN CR was provided in two formats; the received ASTERIX data in a ‘.pcap’ format and the processed data by a program of the Croatia Control in a ‘.xml’ format. The script for converting the data into the required form has been created in a MATLAB version R2017a by using the data import. It may not work for older versions of MATLAB. The data has been converted from the ‘.xml’ format with an application of various delimiters; in particular symbols '<','>' and '/'. Three columns of strings occurred and there was a necessity to filter only required values. A column ‘Data’ contains information which data is saved in the appropriate row of a column ‘Value’. The number of rows is saved into a ‘s’ variable and other variables are padded with the data in a for-loop with loops corresponding to the number of rows. At first, there is testing of an occurrence of a ‘MBdata’ in the ‘Data’ column. If the condition is met, a variable ‘RawMessage’ is filled with a hexadecimal sequence of the BDS register from the appropriate row of the ‘Value’. Subsequently, of a variable ‘l’ and ‘m’ are counted. They correspond to a BDS1 and BDS2 position in the converted data. In other words, the assigned BDS1 code to the correct BDS register is placed one row below the message in ‘Value’ column and the BDS2 code two rows below. The hexadecimal sequence is converted to the binary code with help of the ‘hex2bin2’ function. The converted binary sequence of 56-bits is saved into a variable ‘BDSmessages’ as well as in case of initial data processing of the data from ADS-B receivers. Secondly, the BDS1 code is saved into a ‘BDS1’ variable, on assumption that the value in the ‘Data’ column is set at ‘DSB1’. The same rule is applied at a ‘BDS2’ variable corresponding to a ‘DSB2’ value which is situated two rows under the corresponding message. At last, variables ‘BDS1’ and ‘BDS2’ are connected into one variable ‘BDSasterix’ representing the known BDS number which is crucial for the further evaluation of the heuristic algorithm. The part of the script symbolising this data filtration is specified in Figure 12.

```
%Data filtration
s=size(Data,1); %number of rows
BDS1 = NaN(s,1);
BDS2 = NaN(s,1);
RawMessage = cell(s,1);
MessageBIN = cell(s,1);
BDSmessages=[];
BDSasterix=[];

for k=1:s
    if Data(k,1)== 'MBdata'
       RawMessage{k,1}= Value(k,1);
       Message=RawMessage{k,1};
       l=k+1;
       m=k+2;
       MessageBIN{k,1}=hex2bin2(Message{1,1});
       BDSmessages=[BDSmessages;MessageBIN{k,1}];
        if Data(l,1)=='DSB1'
        BDS1(k,1)= Value(l,1);
            if Data(m,1)=='DSB2'
            BDS2(k,1)= Value(m,1);
            end
        end
        BDSasterix=[BDSasterix;BDS1(k,1) BDS2(k,1)];
    end
end
```

Figure 12. Filtration of required values of ATN data

An output of the initial data processing of the ATN data is saved into a MAT-file. It is named 'AsterixBDS' and it contains variables 'BDSmessages' as well as 'BDSasterix' with the known assigned BDS number. The output is also placed into the 'Test' folder and prepared for using by the heuristic algorithm.

3.2 Data Analysis

Before the creation of the heuristic algorithm for distinguishing particular BDS registers, there has to be a bit analysis processed to find unique bits and combinations of bits which determine a specific register. A significant register part is found according to the message contained in Appendix A for BDS register groups ELS, EHS and MRAR. The 56-bit arrays with known number of BDS register are available for analysis after initial data processing of ATN data. Afterwards, some of the messages have been decoded to find the next criteria to decrease the assignment of more BDS numbers. Due to the fact that a composition of some of the BDS registers are similar, more than one BDS number can be assigned to one message. For that reason, conflicts of these allotments have been analysed and are specified in more detail directly in description of the heuristic algorithm (see Chapter 3.3).

3.2.1 Bit Analysis of the BDS code $1,0_{16}$

On the assumption that a Mode S transponder supports ELS as well as EHS, bit analysis is processed for replies contained information about avionics configuration and operational datalink status. A significant part of BDS code $1,0_{16}$ is situated in the first eight bits of the message. The first four bits (so called BDS1 code) and the second four bits (BDS2 code) must be decoded as hexadecimal value '1' and '0'. Bits on position 17-23 must be equal to decimal 0, 3, 4 or 5 for all Mode S transponders installed under specification after released in 2002 or later. In case that bit on position 25 is set to '0', BDS registers of EHS are not available. BDS code 1,0 shall provide information about Aircraft Identification Capability and Surveillance Identifier Code. Therefore, bits on position 33 and 35 must be set to '1'. On the basis of real data analysis new criteria has been found that bits on position 29-32 are set at zeros in every message. [6] These bits corresponded to information about Downlink ELM: "*throughput capability of downlink ELM containing the maximum number of ELM segments that the transponder can deliver in response to a single requesting interrogation.*" [5]. Options for BDS 10_{16} setting are specified in Table 9 below.

Bit position	Bit setting			
1	0			
2	0			
3	0			
4	1			
5	0			
6	0			
7	0			
8	0			
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0
21	0	0	1	1
22	0	1	0	0
23	0	1	0	1
29-32	0			
33	1			
35	1			

Table 9. BDS code $1,0_{16}$ significant bits

3.2.2 Bit Analysis of the BDS code $1,7_{16}$

On the condition that the Mode S transponder provides ELS and EHS message transmission, appropriate bits of 'Common usage GIBC capability report' shall be set to '1'. It was found that this strict rule excludes many of BDS $1,7_{16}$ messages during the analysis of real data. Therefore, this condition is not used in the algorithm and only one rule for distinguishing this register has been left. This rule is the condition of setting at zero of reserved bits on positions 26-56. Bits are reserved for aircraft capability, Mode S BITE (Built in Test equipment) and military applications. According to real data analysis, this information is not provided and all these bits are set at zero. Information about availability of meteorological data may be found in bits on position 13 for BDS code $4,4_{16}$ and on following position 14 for BDS code $4,5_{16}$. The overview of positions and required setting of the reserved bits are described in Table 10 below. [5] [6]

Bit position	Bit setting
26-56	0

Table 10. BDS code $1,7_{16}$ significant bits

3.2.3 Bit Analysis of the BDS code $2,0_{16}$

A register for aircraft identification can be divided to one group of eight bits and other groups; each of six bits. As with the BDS code $1,0_{16}$, the first part is significant for recognizing register and consists of the first eight bits of the message. In case of Aircraft identification register, the first four bits must be equal to hexadecimal '2' and the second four bits must be set to zeros, as shows Table 11 below. [5] [6]

Bit position	Bit setting
1	0
2	0
3	1
4	0
5	0
6	0
7	0
8	0

Table 11. BDS1 and BDS2 setting for BDS code $2,0_{16}$

Particular characters of the identification are situated in groups of six bits contained in remaining bits of the message. These six bits may represent one of the characters specified in Table 12.

Bit value (decimal)	Corresponding characters
1-26	A-Z
32	Space
48-57	0-9

Table 12. Value options for Aircraft identification

There shall be only bit combinations contained in BDS code $2,0_{16}$ and the character string cannot contain any intervening spaces. Number of 56-bit long BDS register enables input of a maximum of eight characters of aircraft identification. Providing shorter numbers of characters, the unused groups of six bits at the end of the message must be equal to space coding. Any of 6-bit groups of BDS code $2,0_{16}$ are not allowed to be padded with decimal values 27-31, 33-47 or 58-64. [6]

3.2.4 Bit Analysis of the BDS code $3,0_{16}$

Until there an RA (Resolution Advisory) event occurs, bits of BDS code $3,0_{16}$ shall be set to zeros. After the initial load of register 'ACAS active resolution advisory', the first eight bits are set and remains set. BDS1 code must refer 3 and BDS2 code must be set to zeros. The bit sequence in binary code is specified in Table 13 below. [6]

Bit position	Bit setting
1	0
2	0
3	1
4	1
5	0
6	0
7	0
8	0

Table 13. BDS1 and BDS2 setting for BDS code $3,0_{16}$

According to this first part only, it is not possible to distinguish BDS register $3,0_{16}$. Other rules were found in Chapter called 'Field Description' (4.3.8.4.) in Annex 10.IV. At first, there is a condition depending on setting of bit on position 9 representing 'Active RA' (ARA), and bit on position 28 representing 'Multiple Threat Encounter' (MTE). In relation to these bits settings, some values in corresponding field shall be set at zeros. Secondly, if the 'Threat-Type Indicator' (TTI) which is characterized by bits on positions 29 and 30 is set at '1', the two last bits of the message shall be also zeros, see Table 14. [2]

ARA	MTE	Bits set at '0'
1	0	16-22
1	1	16-22
0	1	16-22
0	0	10-22
TTI		Bits set at '0'
0	1	55-56

Table 14 - Significant bits setting for BDS code $3,0_{16}$

3.2.5 Bit Analysis of the BDS code $4,0_{16}$

Enhanced surveillance BDS registers are characterized with particular data sets providing extended flight information to ATM. Availability of each set is specified in an appropriate status bit. On condition that Mode S transponder supports EHS, at least one of bit status must be set to '1'. That means there is vertical information of an aircraft contained in BDS code $4,0_{16}$. In addition, if the status bit is set at '0', following bits that represent corresponding data field must be either padded with zeros. An overview of status bits and their data sets are specified in Table below. [5] [6]

Status bit position	Data field
1	2-13
14	15-26
27	28-39
48	49-51
54	55-56

Table 15. BDS code $4,0_{16}$ status bits

Furthermore, reserved bits in the 'Selected vertical intention' are considered, see Table 16. These bits must be set to '0' and are regarded to another significant bit combination. [5] [6]

Bit position	Bit setting
40-47	0
52-53	0

Table 16. BDS code $4,0_{16}$ reserved bits

3.2.6 Bit Analysis of the BDS code $5,0_{16}$

Providing that EHS is supported and BDS code $5,0_{16}$ contains appropriate data, at least one of the status bit must be set at '1'. If a status bit is padded with '0', corresponding data field does not contain any flight information and there shall be situated a bit sequence of zeros in the field. BDS register $5,0_{16}$ is divided into five data sets, their status bits are specified in Table 17. Contrary to 'Selected vertical intention', this report does not contain any reserved bits. [5] [6]

Status bit position	Must be set '0' (bit positions)
1	2-11
12	13-23
24	25-34
35	36-45
46	47-56

Table 17. BDS code $5,0_{16}$ status bits

3.2.7 Bit Analysis of the BDS code $6,0_{16}$

Such as previous BDS register $5,0_{16}$, 'Heading and speed report' is divided into five data sets represented by status bit. The only difference is a position of one status bit and associated length of two data fields. The same message padding is applied to BDS code $6,0_{16}$ as to previous EHS. As well as 'Track and turn report', BDS register $6,0_{16}$ does not contain any reserved bits. Its status bits are specified in Table 18 below. [5] [6]

Status bit position	Must be set '0' (bit positions)
1	2-12
13	14-23
24	25-34
35	36-45
46	47-56

Table 18. BDS code $6,0_{16}$ status bits

3.2.8 Bit Analysis of the BDS code $4,4_{16}$

However, four bits in the beginning of BDS code $4,4_{16}$ differs from EHS registers, the remaining part of the message is also characterized with status bits for four data fields. The first part of bit sequence represents specific data resource and can be set to binary codes described in Table 19. Other bit combinations are considered reserved, therefore there is not assumed their occurrence in 'FOM/SOURCE' field. [5]

Bit position	Bit setting				
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	1	1	0
4	0	1	0	1	0

Table 19. FOM/SOURCE data set decoding

As mentioned before, BDS code $4,4_{16}$ is characterized by status bits. Obviously, there is the same rule for status bit and following field like in previous registers. The overview of 'Meteorological routine air report' status bit can be found in Table 20 below.

Status bit position	Must be set '0' (bit positions)
5	6-23
23	24-34
35	36-46
47	48-49
50	51-56

Table 20. BDS code $4,4_{16}$ status bits

3.2.9 Bit Analysis of the BDS code $4,5_{16}$

Status bits of 'Meteorological hazard report' are in most cases followed by bit pairs representing a value of relevance of dangerous meteorological phenomena. This information is complemented by routine meteorological information, which can be found also in BDS code $4,4_{16}$ (static air temperature and average static pressure data fields). Eight status bits are significant for BDS code $4,5_{16}$ and they are further characterised in Table 21. [5]

Status bit position	Must be set '0' (bit positions)
1	2-3
4	5-6
7	8-9
10	11-12
13	14-15
16	17-26
27	28-38
39	40-51

Table 21. BDS code $4,5_{16}$ status bits

What is more, there is another significant part of the message in the end of the BDS code $4,5_{16}$. A sequence of the last five bits are considered reserved for this reason these bit values shall be set at '0', see Table 22.

Bit position	Bit setting
52-56	0

Table 22. BDS code $4,5_{16}$ reserved bits

3.3 Heuristic Algorithm

After the data filtration, only 56-bit binary arrays of a binary sequence are available for further processing. They represent contents of BDS registers from received messages with DF 20 or DF 21 in a binary code. The bit analysis has been processed where unique combinations of the binary code have been found. These unique parts of the code characterize particular BDS registers and cannot be contained in any other register. Together with other criteria,

the heuristic algorithm can proceed a testing of specific parts and can assign a number of BDS register to the binary sequence when all required conditions are met.

The algorithm testing has been divided into three parts:

- The testing of unique parts and other specific criteria;
- The BDS number assignment according to the testing;
- The final saving of assigned BDS messages.

An input for the algorithm can be represented by any 56-bit binary sequence containing a Mode S transponder reply on a SSR interrogation. In the case of this thesis, the input is specified by provided data by ATN CR or by received data from ADS-B receivers operated by the laboratory. This data is loaded from 'Test' folder after its initial processing. Initially, some variables are defined for using within the algorithm. A bit array with the tested message is saved into a variable 'Tested_message'. In order to use the variable 'Tested_message' not only in a local workspace, but also as a function, this variable is declared as global. All other variables are local variables and they are saved only into the workspace of the script. A number of messages is characterized by a number of rows which is counted in an 's' variable. All available opportunities of a BDS number assignment are specified in a variable 'BDS_number_guess' which contains nine rows corresponding to the number of assigned BDS registers types. These estimated numbers are further examined and thanks to a variable 'BDS_guess_counter' and other conditions only one final BDS number can be assigned to one message which is finally saved into a 'BDS_final'. The initial defining is represented in Figure 13.

```
%defining variables
global Tested_message; %global variable for used function
Tested_message=[]; %defining global variable
s=size(BDSmessages,1); %number of BDS rows
BDS_number_guess=zeros(s,9); %saving of BDS number, distinquishing of 9 different registers
BDS_guess_counter=zeros(s,1); %counting assign register number
BDS_final=cell(s,1); %final BDS register number
```

Figure 13. Initial variables defining

A required output of the algorithm is to assign the appropriate BDS number to the tested message. This is ensured by the 'BDS_final' variable. There is a number of the assigned BDS register on a row corresponding to the row of the tested message. Moreover, the particular BDS registers from a data sample are saved into variables with appropriate name. The number of the register is expressed by a simple number instead of the typical notation; for example the 'BDS_final' is padded with 40 when the message is considered

BDS code $4,0_{16}$. All original messages of distinguished BDS registers are saved into a MAT-file. In addition, meteorological codes $4,4_{16}$ and $4,5_{16}$ are saved into an another MAT-file, which is further used for decoding these messages and obtaining meteorological information from onboard an aircraft.

The whole Matlab script called 'Heuristic_algorithm.m' is specified in the Appendix D below.

3.3.1 Testing of Unique Parts and Other Specific Criteria

A for-loop provides processing of a particular message where the message is tested by every possible condition to assign or exclude an assignment of all possible BDS numbers. The tested message is saved into the global variable 'Tested_message' and if there is at least one bit set at '1', the message is further examined, as specified in Figure 14.

```
%BDS assigning criteria testing
for i=1:s;
  Tested_message=BDSmessages(i,:); %tested message

  %Exluding zeros messages
  ZeroCounter=sum(BDSmessages(i,1:56)==0); %number of zeros in a message
  if ZeroCounter(1,1)<=55; %no testing of zeros messages
```

Figure 14. Excluding of zeros messages

The whole process can be divided into three parts. At first, there is a distinguishing part in the beginning of the bit sequence which is represented by the first four bits (BDS1) and second four bits (BDS2). There must be decoded the numbers corresponding to the BDS designation in these bits. For example, BDS1 must be set at '0010' and BDS2 must be set at zeros to identifying of BDS code $2,0_{16}$. This rule can be applied only for BDS code $1,0_{16}$, BDS code $2,0_{16}$ and BDS code $3,0_{16}$. Any other register does not contain the BDS1 or the BDS2 code in the beginning of the binary code. The second part is represented by testing a BDS code $1,7_{16}$. This register cannot be identified by BDS1 and BDS2 code neither it is characterized by any status bits. The testing which confirms or disproves the final number of this BDS code is made on specific bits, which are stated as reserved and numbers of ones and zeros in the message. The third group of registers is represented by status bits. A basic condition for this group is that if a status bit is set at '0', the following bits with a required information must be also set at '0'. This step repeats at more BDS registers, therefore, a MATLAB function 'Decoding.m' is created and used for testing, see Figure 15. The function is applied to all three BDS registers of EHS (the BDS code $4,0_{16}$, BDS code $5,0_{16}$ and BDS code $6,0_{16}$) as well as to two meteorological registers (the BDS code $4,4_{16}$ and $4,5_{16}$).

```
function [Output] = Decoding(StatusBit,FieldStart,FieldEnd)

Decode=[]; %zero decoding
Decimal=[]; %saving variable

global Tested_message;
for a=1:size(StatusBit,2); %all status fields
    Decode(1,a)=bi2de(Tested_message(1,FieldStart(1,a):FieldEnd(1,a)),'left-msb');
    if (StatusBit(1,a)==0 && Decode(1,a)==0) || StatusBit(1,a)==1; %Zero condition
        Decimal(1,a)=Decode(1,a);
    else
        Decimal(1,a)=999999;
    end
end
Output=Decimal;
end
```

Figure 15. 'Decoding.m' function

The first BDS registers which are tested are characterized with the sequence of first eight bits in the beginning of the message. A variable 'BDSidentifier' represents the significant part in the beginning of the bit sequence; first four bits (BDS1), and the second four bits (BDS2). The bits are decoded by using a 'bi2de' function with specifying of the most significant bit on the left side of the sequence. Then, the testing of a unique part may occur.

In the first step of 'Data link capability report' identifying is tested whether the 'BDSidentifier' is equal to sixteen, which coincides with converting decimal '10' to hexadecimal scale. Subsequently, significant bits, on positions 17-23, 21-23 and 29-32, are decoded into a decimal number. There have to be required values in these fields which are consequently tested by an if-condition. The new criteria for the BDS code $1,0_{16}$ was found in the real operation data. The bits on position 33 and 35 were always set at '1' and there shall be more than 43 bits set at '0'. If all the conditions are met, the messages may be considered BDS code $1,0_{16}$ and a corresponding row of 'BDS_number_guess' is padded with '10' in the first column. The testing is described in Figure 16 below.

```
%Testing registers with BDS1 and BDS2 code
BDSidentifier=bi2de(BDSmessages(i,1:8),'left-msb'); %saving BDS1 and BDS2 code in decimal scale

  %BDS 1,0
  if BDSidentifier==16; %hexadecimal 1,0 in decimal scale
      BDSOneZeroGuess=BDSmessages(i,:); %saving for following testing
      OneZeroField(1,1)=bi2de(BDSOneZeroGuess(17:20),'left-msb'); %reserved values in field
      OneZeroField(1,2)=bi2de(BDSOneZeroGuess(21:23),'left-msb'); %required values in field
      OneZeroField(1,3)=bi2de(BDSOneZeroGuess(29:32),'left-msb'); %reserved values in field
          if OneZeroField(1,1)==0 && (OneZeroField(1,2)==0 || OneZeroField(1,2)==3 || OneZeroFie
             BDS_number_guess(i,1)=10; %BDS number saving
          end
```

Figure 16. BDS1 and BDS2 decoding and testing of BDS code $1,0_{16}$

On the condition that BDS1 is set at hexadecimal ‘2’ and BDS2 at ‘0’, the ‘BDSidentifier’ is equal to thirty-two in decimal scale, and the appropriate message is saved into a ‘BDSTwoZeroGuess’ variable with considering that this row may be message with ‘Aircraft identification’. The BDS register $2,0_{16}$ is further divided into groups of six bits contained in remaining bits of the message, specifying particular characters of the identification. The characters are decoded into decimal values represented by other variable ‘IDField’. These fields are tested whether they do contain any of the decimal values which are not allowed there. Undefined values are saved in a ‘NoDefined’ vector and tested in a for-loop for every field. When the ‘BDSidentifier’ is equal to thirty-two, and there is no field containing an undefined value, the row with received message is considered to be the BDS code $2,0_{16}$ and the second column of corresponding row of ‘BDS_number_guess’ is set at ‘20’. The distinguishing of the BDS register $2,0_{16}$ is specified in Figure 17 below.

```
%BDS 2,0
elseif BDSidentifier==32; %hexadecimal 2,0 in decimal scale
        BDSTwoZeroGuess=BDSmessages(i,:); %saving for other testing
        IDField(1,1)=bi2de(BDSTwoZeroGuess(1,9:14),'left-msb'); %decoding characters field
        IDField(1,2)=bi2de(BDSTwoZeroGuess(1,15:20),'left-msb');
        IDField(1,3)=bi2de(BDSTwoZeroGuess(1,21:26),'left-msb');
        IDField(1,4)=bi2de(BDSTwoZeroGuess(1,27:32),'left-msb');
        IDField(1,5)=bi2de(BDSTwoZeroGuess(1,33:38),'left-msb');
        IDField(1,6)=bi2de(BDSTwoZeroGuess(1,39:44),'left-msb');
        IDField(1,7)=bi2de(BDSTwoZeroGuess(1,45:50),'left-msb');
        IDField(1,8)=bi2de(BDSTwoZeroGuess(1,51:56),'left-msb');
        NoDefined=[0 27:31 33:47]; %not allowed values
        for l=1:length(NoDefined);
            if IDField (1,:)~= NoDefined(l); %testing of allowed values
                if IDField (1,1)~=32 || IDField (1,2)~=32 || IDField (1,3)~=32 || IDField (1,4)
                BDS_number_guess(i,2)=20; %BDS number saving
                end
            end
        end
```

Figure 17. Sample of testing BDS code $2,0_{16}$

The ‘BDSidentifier’ testing is processed for distinguishing the BDS code $3,0_{16}$. When a group of eight bits set at ‘00110000’ is converted to the decimal scale, the result is equal to forty-eight. Four fields are decoded into a decimal field and on the condition that bit on a position 9 and bit on a position 28 are set at a specific combination some of the defined fields shall be padded with zeros. Number ‘30’ is filled into the appropriate row and third column of the ‘BDS_number_guess’ when required conditions are met. The ‘ACAS active resolution advisory’ testing is shown in Figure 18.

```
%BDS 3,0
elseif BDSidentifier==48; %hexadecimal 3,0 in decimal scale
    ThreeZeroField(1,1)=bi2de(BDSmessages(i,16:22),'left-msb');
    ThreeZeroField(1,2)=bi2de(BDSmessages(i,10:22),'left-msb');
    ThreeZeroField(1,3)=bi2de(BDSmessages(i,55:56),'left-msb');
    ThreeZeroField(1,4)=bi2de(BDSmessages(i,29:30),'left-msb');
    if (BDSmessages(i,9)==1 && BDSmessages(i,28)==0) || (BDSmessages(i,9)==1 && BDSmessages(i,28)
        if ThreeZeroField(1,4)==1 && ThreeZeroField(1,3)==0;
            BDS_number_guess(i,3)=30;
        end
    elseif BDSmessages(i,9)==0 && BDSmessages(i,28)==0 && ThreeZeroField(1,2)==0;
        if ThreeZeroField(1,4)==1 && ThreeZeroField(1,3)==0;
            BDS_number_guess(i,3)=30;
        end
    end
end
```

Figure 18. Testing of BDS code 3,0$_{16}$

The next part of the identification of BDS registers focuses on testing of 'Common usage GIBC capability report'. The condition was considered that if a Mode S transponder provides ELS and EHS message transmission, bits on position 7, 9, 16 and 24 shall be set at '1'. However, it was excluded after examining these criteria in the real operation data. It reported some of the bits are set at '0' and many messages was not assigned like the BDS code 1,7$_{16}$, although it would have been the correct allocation. Only a reserved bits check and number of zeros and ones remained by the testing of this register. The bits are reserved for military applications, but there were not any of the bits set at '1' within the data from ATN CR. A corresponding row in the ninth column is filled with '17' in the 'BDS_number_guess' variable, when reserved bits are padded with '0' and the message contains more than forty zeros, alike in Figure 19.

```
%BDS 1,7
OneSevenField(1,1)=bi2de(BDSmessages(i,26:56),'left-msb'); %reserved bits
if OneSevenField(1,1)==0 && ZeroCounter(1,1)>=40; %conditions for BDS 1,7
BDS_number_guess(i,9)=17; %BDS number saving
end
```

Figure 19. Condition testing for BDS code 1,7$_{16}$

The last section of the identification is determining BDS registers of EHS (BDS code 4,0$_{16}$, BDS code 5,0$_{16}$ and BDS code 6,0$_{16}$), as well as meteorological registers (BDS code 4,4$_{16}$ and 4,5$_{16}$). There are three necessary variables defined for each of the registers; a variable with setting of status bit, a vector of the first bit position of an appropriate field and a vector with the bit position in the end of the field. An example of defining the variables for the BDS code 4,0$_{16}$ is specified in Figure 20.

```
%Defining variables for EHS and MRAR registers
%significant bits for BDS 4,0
FourZeroStatusBit=[BDSmessages(i,1) BDSmessages(i,14) BDSmessages(i,27) BDSmessages(i,48)
FourZeroStartBit=[2 15 28 49 55];
FourZeroEndBit=[13 26 39 51 56];
```

Figure 20. Example of defining variables for BDS 4,0$_{16}$

Status bits, and positions of the first and last bits of fields are set up according to the number of tested fields and bit analysis.

The first tested register of EHS is the 'Selected vertical intention'. Initially, reserved fields and a field with a required value is converted to the decimal scale and a number of ones in the tested message is counted. The first field with bits on position 29-31 represents the beginning of bit sequence characterizing an information about barometric pressure. There may not be a decoded value higher than 2220, which means that these bits shall be set at '0' in every BDS code 4,0$_{16}$. These conditions are tested on purpose of increasing the algorithm effectivity; to shorten time of the testing, and to exclude other decoding and testing of the message, see Figure 21.

```
%BDS 4,0
FourZeroField(1,1)=bi2de(BDSmessages(i,29:31),'left-msb'); %required values in field
FourZeroField(1,2)=bi2de(BDSmessages(i,40:47),'left-msb'); %reserved bits
FourZeroField(1,3)=bi2de(BDSmessages(i,51:53),'left-msb'); %reserved bits
FourZero_OneCounter(1,1)=sum(BDSmessages(i,1:56)==1); %number of ones
    if FourZeroField(1,1)==0 && FourZeroField(1,2)==0 && FourZeroField(1,3)==0 && FourZero_One
       DecodedFourZeroGuess=Decoding(FourZeroStatusBit,FourZeroStartBit,FourZeroEndBit); %stat
        if DecodedFourZeroGuess(1,1)<=2950; %maximum altitude
            if DecodedFourZeroGuess(1,2)<=2950; %maximum altitude
                if DecodedFourZeroGuess(1,3)==0 || (DecodedFourZeroGuess(1,3)<=2220 && Decoded
                    if DecodedFourZeroGuess(1,4)==0 || DecodedFourZeroGuess(1,4)==2 || Decoded
                        if DecodedFourZeroGuess(1,5)<=3; %possible acquired values
                    BDS_number_guess(i,4)=40; %BDS number saving
                        end
                    end
                end
            end
        end
    end
```

Figure 21. BDS register 4,0$_{16}$ testing

If the initial condition is met, there is a test using 'Decoding.m' function whether the condition of zero-status bit is met, and a result is put in a 'DecodedFourZeroGuess' variable. This variable consists of five fields according to a construction of the register and the previous defining. The fields contain decoded decimal values or a value '999999' when the status bit condition is not met. A test is performed, if the reserved fields are set to zeros and a testing of some of the decoded values in 'DecodedFourZeroGuess'. The limitation of the values is represented by Table 23.

Field Name	Limitation Meaning	Decimal Limitation
MPC/FCU Selected Altitude	47200 ft	2950
FMS Selected Altitude	47200 ft	2950
Barometric Pressure	0, 1008-1022 mb	0, 2080-2220
MCP/FCU Mode	Approach mode, Alt hold mode, VNAV mode	0, 2, 4
Target Alt Source	No limitation	3

Table 23. BDS code $4,0_{16}$ values limitation

Values of limitation for significant fields are determined by analysis of the real operation data, which was decoded and the range of the field content was implemented into the algorithm. Every field shall be checked because of the zero-status bit condition. It ensures that any of the inadequate messages are excluded for the BDS number assignment. If there is no field containing non-zero bits if their status bit is set at '0', reserved fields are set to zero, and the limitation in fields is not exceeded, the row of BDS_number_guess is padded with a value '40' in the fourth column.

Similarly, a BDS register $5,0_{16}$ testing is processed. Before the decoding, some significant bits from the beginning of specific fields are tested to exclude testing of a large number of messages. Thanks to the 'Decoding.m' function, the row is saved into 'DecodedFiveZeroGuess' variable where the zero-status condition is met. Significant fields from this variable are tested whether the decimal value does not exceed any of the limitations specified in Table 24 below. In comparison with BDS $4,0_{16}$, there are no reserved fields in the 'Track and turn report'.

Field Name	Limitation Meaning	Decimal Limitation
Roll angle	0-30°, -30°-0°	0-170, 850-1024
True track angle	No limitation	2048
Ground speed	540 kn	270
Track angle rate	8°/s	1024
True airspeed	540 kn	270

Table 24. BDS code $5,0_{16}$ values limitation

The field containing information about true track angle may take values within the full range of the field, but there shall still be a testing whether the value does not exceed '999999'. On condition, that rows do not contain any status bit set at '0' followed by a non-zeros field, and the limitation in fields is not exceeded, the appropriate row of 'BDS_number_guess' is filled with value '50' in the fifth column. There is an example of assigning BDS number of this EHS register in Figure 22.

```
%BDS 5,0
if BDSmessages(i,25)==0 && BDSmessages(i,47)==0 && BDSmessages(i,48)==0; %testing of the most
   DecodedFiveZeroGuess=Decoding(FiveZeroStatusBit,FiveZeroStartBit,FiveZeroEndBit); %status
    if DecodedFiveZeroGuess(1,1)<=170 || (DecodedFiveZeroGuess(1,1)>=850 && DecodedFiveZeroGu
        if DecodedFiveZeroGuess(1,2)<=2048; %true track angle
            if DecodedFiveZeroGuess(1,3)<=270; %maximum ground speed
                if DecodedFiveZeroGuess(1,4)<=1024; %maximum track angle
                    if DecodedFiveZeroGuess(1,5)<=270; %maximum true airspeed
                       BDS_number_guess(i,5)=50; %BDS number saving
                    end
                end
            end
        end
    end
end
```

Figure 22. BDS register $5,0_{16}$ testing

The only difference between the BDS register $5,0_{16}$ and the BDS register $6,0_{16}$ is a position of the second status bit, which means that also the second field has another border bits positions. Likewise, the testing of BDS code $5,0_{16}$, some significant bits are tested in the beginning. Then, particular fields are decoded and their values are checked, whether they exceed any of the limitations, specified in Table 25 below.

Field Name	Limitation Meaning	Decimal Limitation
Magnetic heading	No limitation	2048
Indicated airspeed	540 kn	540
Mach number	0.92 M	230
Barometric altitude rate	0-4160, -4160-0 ft/min	0-130, 884-1024
Inertial vertical velocity	0-4092, -4092-0 ft/min	0-124, 900-1024

Table 25. BDS code $6,0_{16}$ values limitation

In case, all conditions of the BDS $6,0_{16}$ testing are met, a value ‘60’ is saved into sixth column of corresponding row with the ‘Heading and speed report’ in the variable ‘BDS_number_guess’. The specific part of the script with this BDS code determining is represented in Figure 23.

```
%BDS 6,0
if BDSmessages(i,14)==0 && BDSmessages(i,25)==0 && BDSmessages(i,26)==0 && BDSmessages(i,35)==1;
   DecodedSixZeroGuess=Decoding(SixZeroStatusBit,SixZeroStartBit,SixZeroEndBit); %status bit tes
   if DecodedSixZeroGuess(1,1)<=2048; %magnetic heading
       if DecodedSixZeroGuess(1,2)<=540; %maximum indicated airspeed
           if DecodedSixZeroGuess(1,3)<=230; %maximum speed
               if DecodedSixZeroGuess(1,4)<=130 || (DecodedSixZeroGuess(1,4)>=884 && DecodedSixZ
                   if DecodedSixZeroGuess(1,5)<=124 || (DecodedSixZeroGuess(1,5)>=900 && Decoded
                   BDS_number_guess(i,6)=60; %BDS number saving
                   end
               end
           end
       end
    end
end
```

Figure 23. BDS register 6,0$_{16}$ testing

In case of 'Meteorological Air Routine Report', the first four bit are tested, where the value of FOM/SOURCE field are saved. In this field only values 1-4 are allowed. The status bit condition is tested by the 'Decoding.m' function and the corresponding row of 'DecodedFiveZeroGuess' variable is filled, when the condition is met. Afterwards, all values in the decoded field are checked, if they do not exceed any limitation, see Figure 24.

```
%BDS 4,4
FourZeroField(1,1)=bi2de(BDSmessages(i,1:4),'left-msb'); %FOM/SOURCE field decoding
   if (FourZeroField(1,1)==1 || FourZeroField(1,1)==2 || FourZeroField(1,1)==3 || FourZeroField(
      DecodedFourFourGuess=Decoding(FourFourStatusBit,FourFourStartBit,FourFourEndBit); %status
       if DecodedFourFourGuess(1,1)<=80; %maximum wind speed
           if DecodedFourFourGuess(1,2)<=512; %wind direction
               if DecodedFourFourGuess(1,3)<=240 || (DecodedFourFourGuess(1,3)>=704 && DecodedFo
                   if DecodedFourFourGuess(1,4)<=1080; %average static pressure
                       if DecodedFourFourGuess(1,5)<=3; %turbulence status
                           if DecodedFourFourGuess(1,6)<=64; %maximum humidity
                               BDS_number_guess(i,7)=44; %BDS number saving
                           end
                       end
                   end
               end
           end
       end
   end
```

Figure 24. BDS register 4,4$_{16}$ testing

The range of the static air temperature is specified in the Technical Provisions for Mode S Services and Extended Squitter [3]. The wind speed was adjusted according to decoded values of the real operation data. The last two fields containing information about average static pressure and humidity are rarely padded with bits. If the condition of allowed setting of first four bit is met and the limitation is not exceeded, the seventh column of corresponding row is padded with '44' into the 'BDS_number_guess'. Limitations for the BDS code 4,4$_{16}$ are specified in Table 26.

Field Name	Limitation Meaning	Decimal Limitation
Wind speed	80 kn	80
Wind direction	No limitation	512
Static air temperature	0-60, -80-0°C	0-240, 704-1024
Average static pressure	1080 hPa	1080
Turbulence	No limitation	3
Humidity	100%	64

Table 26. BDS code $4,4_{16}$ values limitation

A 'Meteorological hazard report' testing is prepared for using in the heuristic algorithm, in case that this message will be transmitted in the future. The BDS code $4,5_{16}$ is tested only on condition that any other BDS number has not been assigned during the previous testing. There is low probability of occurrence of hazardous phenomena with the 'severe' interpretation. Therefore, the number of bits set at '1' is tested and shall not be higher than ten. Not only reserved bits are examined in the beginning of the remaining message testing, but also a number of ones within hazardous phenomena. Then, all eight fields of BDS code $4,5_{16}$, are tested and decoded by the 'Decoding.m'. The eighth column of the variable 'BDS_number_guess' is reserved for filling in with '45', in case all conditions and limitations suits, see Figure 25.

```
%BDS 4,5
    if BDS_number_guess(i,:)==0; %testing only if no other BDS number has been assigned
       FiveZeroField(1,1)=bi2de(BDSmessages(i,52:56),'left-msb'); %reserved field decoding
       OnesValues=sum(BDSmessages(i,1:15)==1); %hazardous values set at 1 counting
       ZerosValues=sum(BDSmessages(i,1:15)==0); %hazardous values set at 0 counting
           if FiveZeroField(1,1)==0 && OnesValues<=10 && OnesValues>=4; %reserved field and
              DecodedFourFiveGuess=Decoding(FourFiveStatusBit,FourFiveStartBit,FourFiveEndB
               if DecodedFourFiveGuess(1,1)<=3; %turbulence
                   if DecodedFourFiveGuess(1,2)<=3; %wind shear
                       if DecodedFourFiveGuess(1,3)<=3; %microburst
                           if DecodedFourFiveGuess(1,4)<=3; %icing
                               if DecodedFourFiveGuess(1,5)<=3; %wake vortex
                                   if DecodedFourFiveGuess(1,6)<=240 || (DecodedFourFiveGues
                                       if DecodedFourFiveGuess(1,7)<=1080 && DecodedFourFive
                                           if DecodedFourFiveGuess(1,8)<=2950 && DecodedFou
                                              BDS_number_guess(i,8)=45; %BDS number saving
                                           end
                                       end
                                   end
                               end
                           end
                       end
                   end
               end
           end
    end
```

Figure 25. BDS register $4,5_{16}$ testing

Hazardous phenomena may acquire decimal values from zero to three. In the other words, any of tested bit combinations is included into this limitation. Nevertheless, there still shall be the value testing for excluding messages which do not met the zero-status bit condition. Limitations of of BDS register $4,5_{16}$ are tested, which is represented in Table 27 below.

Field Name	Limitation Meaning	Decimal Limitation
Turbulence	No limitation	3
Wind shear	No limitation	3
Microburst	No limitation	3
Icing	No limitation	3
Wake vortex	No limitation	3
Static air temperature	0-60, -80-0°C	0-240, 704-1024
Average static pressure	1080 hPa	1080
Radio height	47200 ft	2950

Table 27. BDS code $4,5_{16}$ values limitation

The whole for-loop for the messages testing is enclosed with counting of assigned BDS numbers, which is used in the following step.

3.3.2 BDS Number Assignment According to the Testing

The main reason for creating of this part of the algorithm is to assign the right BDS number to the tested message. A situation may occur that more than one BDS number is estimated to a message. This collision shall be resolved and the final BDS number is saved into the 'BDS_final' variable.

There is an another for-loop created with the cycle including all messages from the tested sample of data. Based on the number of guessed possibilities, final values are directly filled into the 'BDS_final' or the messages are further tested to assign the appropriate BDS number. The direct BDS number assignment is processed by messages where no BDS number was estimated, on assumption that this BDS register is not included in the group of tested BDS codes. If there is more than one estimated BDS number by the BDS code $1,0_{16}$ or the BDS $2,0_{16}$, there will be always direct padding the 'BDS_final' variable with the corresponding number, because the previous testing has been considered sufficient for this pair of ELS registers. The last direct assignment is applied when there is only one guessed BDS number, with exception of the BDS register $4,5_{16}$. While this register is not transmitted, the messages with assigned its number are marked as 'Unknown'. The described conditions for final determining the BDS number is characterized in Figure 26.

```
%final BDS number assignment
for i=1:s;
    %Unknown BDS
    if BDS_guess_counter(i,1)==9; %another BDS register
        BDS_final{i,1}='Unknown'; %final BDS number assignment

    %BDS with sufficient testing (BDS 1,0 and BDS 2,0)
    elseif BDS_number_guess(i,1)==10;
        BDS_final{i,1}=10; %final BDS number assignment

    elseif BDS_number_guess(i,2)==20;
        BDS_final{i,1}=20; %final BDS number assignment

    elseif BDS_guess_counter(i,1)==8; %only one assigned BDS number
        for j=1:9;
            if BDS_number_guess(i,j)~=0; %choosing the one assigned BDS number
                BDS_final{i,1}=BDS_number_guess(i,j); %final BDS number assignment
            end
        end
        if BDS_number_guess(i,8)==45; %BDS 45 is not trasmitted
            BDS_final{i,1}='Unknown';
        end
```

Figure 26. Direct BDS number assignment

When a conflict of two estimated BDS numbers occurred, there was a data analysis processed to find out, whether there are any other conditions to be explored to do a final decision. Any additional condition was not needed in most of the cases and the double assignments had been solved as the final allocation of one of the registers. In case of estimated both, 40 or 50, there was no BDS register $5{,}0_{16}$ within real operation data, as well as in conflict between 44 or 17, and 50 or 17. When a collision of BDS register $5{,}0_{16}$ with $6{,}0_{16}$ occurred, only 0,57% of the messages were considered BDS code $5{,}0_{16}$ according to the ATN data. A decision, whether the message is 40 or 60, depends on the number of bits set at '1' in the messages. If the number is equal or less than fifteen, the message is regarded BDS register as BDS register $4{,}0_{16}$. A difference between BDS code $4{,}0_{16}$ and $1{,}7_{16}$ creates the testing bits on specific position and number of ones in message, as shown in Figure 27.

```
%40 and 17 conflict
if BDS_number_guess(i,4)==40 && BDS_number_guess(i,9)==17;
    OnesCounter=sum(BDSmessages(i,1:56)==1); %number of ones
    if OnesCounter<=2 && BDSmessages(i,7)==1 && BDSmessages(i,16)==0;
        BDS_final{i,1}=17; %final BDS number assignment
    else BDS_final{i,1}=40; %final BDS number assignment
    end
end
```

Figure 27. An example of collision between two BDS numbers resolution

Two collisions were identified where three BDS numbers were assigned to one message. In the other words, a 'BDS_guess_counter' variable was equal to 6. A decision of the final

BDS number among BDS registers $4,0_{16}$, $5,0_{16}$ and $1,7_{16}$ is made on basis of counting bits set at one. And in the second case, where is a conflict among BDS registers $4,0_{16}$, $4,4_{16}$ and $1,7_{16}$, the final assignment to the register is resolved as 'Unknown', see Figure 28.

```
elseif BDS_guess_counter(i,1)==6; %three assigned BDS registers

    %40, 50 and 17 conflict
    if BDS_number_guess(i,4)==40 && BDS_number_guess(i,5)==50 && BDS_number_guess(i,9)==17;
        OnesCounter=sum(BDSmessages(i,1:56)==1); %number of ones
        if OnesCounter>=4;
        BDS_final{i,1}=17; %final BDS number assignment
        else BDS_final{i,1}='Unknown';
        end
    end

    %40, 44 and 17 conflict
    if BDS_number_guess(i,4)==40 && BDS_number_guess(i,7)==44 && BDS_number_guess(i,9)==17;
        BDS_final{i,1}='Unknown'; %final BDS number assignment
    end
```

Figure 28. Solution of three BDS numbers assignment

Any other conflicts have not happened within the data processing from receivers and from ATN CR. Therefore, any other testing is unnecessary within the algorithm.

3.3.3 Final Saving of Assigned BDS Messages

The saving of the algorithm result is provided for an overview of all assigned messages as well as for further possible decoding of particular groups of the registers, when the higher amount of data is tested. It does not need to be used in specific case, if only one message is tested within the other process, and so on. This part may seem to be complicated, but it was adapted to increase the effectivity of the algorithm.

Initially, empty variables are defined to count number of assigned messages for all groups of the BDS registers. The number of messages assigned to each group are counted in the following for-loop. Afterwards, variables for saving messages are created with appropriate number of rows according to the previous counting, as represented in Figure 29.

```
%defining variables for saving
BDSOneZero=zeros(a,56);
BDSTwoZero=zeros(b,56);
BDSThreeZero=zeros(c,56);
BDSFourZero=zeros(d,56);
BDSFiveZero=zeros(e,56);
BDSSixZero=zeros(f,56);
BDSFourFour=zeros(g,56);
BDSFourFive=zeros(h,56);
BDSOneSeven=zeros(j,56);
```

Figure 29. Defining variables with appropriate size

The initially defined variables for counting number of messages redefined again and their content is set at ‘1’. It enables to fill a binary sequence into an appropriate group messages according to the assigned final BDS number. For example, all messages with the final BDS number ‘40’ are included in a ‘BDSFourZero’ variable in the last for-loop of this script. After the inclusion, one is added to a counting variable to fill the following row in the next loop, as characterized in Figure 30.

```
%saving corresponding messages
for i=1:s;
    if BDS_final{i,1}==10;
        BDSOneZero(a,:)=BDSmessages(i,:);
        a=a+1;
    elseif BDS_final{i,1}==20;
        BDSTwoZero(b,:)=BDSmessages(i,:);
        b=b+1;
    elseif BDS_final{i,1}==30;
        BDSThreeZero(c,:)=BDSmessages(i,:);
        c=c+1;
```

Figure 30. Including messages to their corresponding groups

Whenever it is necessary, these variables may be saved into a MAT-file. By way of illustration, there were two MAT-files saved into ‘Meteo’ folder, one containing variables 'BDSFourFour' and 'BDSFourFive', and the 'AsterixIdentifiedBDS' MAT-file with all groups of distinguished BDS registers. The possible saving is presented in Figure 31.

```
%saving all registers to MAT-file
name=('AsterixIdentifiedBDS');
file=['.\Meteo\',name,'.mat'];
save(file,'BDSOneZero','BDSTwoZero','BDSThreeZero','BDSFourZero','BDSFiveZero','BDSSixZero',

%saving MRAR registers to MAT-file
name=('AsterixMeteoBDS');
file=['.\Meteo\',name,'.mat'];
save(file,'BDSFourFour','BDSFourFive');
```

Figure 31. Saving identified BDS registers into MAT-file

4 Algorithm Functionality Check

The functional check of the created heuristic algorithm has been processed on a sample of data obtained from ADS-B receivers operated by the Department of Air Transport. This check provides information whether the algorithm is able to assign a BDS number to tested messages which is considered the main function of the algorithm. The success rate of particular assignments is further specified for every type of tested BDS register in the following chapter (Chapter 5).

There was a sample of data chosen from 1th of August 2017 for the algorithm check. The data was downloaded in '.dat' format where were all types of messages transmitted on frequency 1090 MHz received. At first, the original sample '1708010200.dat' had to be processed by the initial processing script 'Data_initial_processing_ADSB.m' to filter only required parts and types of received messages. Afterwards, 56-bit sequences of binary code containing messages with DF 20 or 21 remained and were saved into a 'BDSmessages' variable which represented a MAT-file '1708010200.m', see Figure 32.

BDSmessages <210754x56 double>

	1	2	3	4	5	6	7	8	9	10	11
1	1	0	0	0	0	0	0	0	0	0	1
2	1	0	1	0	1	1	1	1	1	1	0
3	1	0	0	0	0	0	0	0	0	0	1
4	1	1	1	0	1	1	0	0	0	0	1

Figure 32. The variable 'BDSmessages'

The MAT-file '1708010200.m' contains more than two hundred thousand rows padded with zeros and ones. This file was loaded into the heuristic algorithm and so the functionality check was processed. A BDS number or a 'Unknown' symbol was assigned to each of tested messages. On the basis of results of the 'BDS_final' variable as well as the 'BDS_number_guess' variable a statistics was created.

First of all, a number of messages containing only zeros within their 56 bits were counted. This number represents approximately 2% of all examined bit sequences. The number of zeros rows is further deducted from a total number of tested messages. These messages were indicated as 'Unknown' and were not included into the statistics calculations. An overview of particular numbers is presented in Table 28.

Category	Number
Total messages	**210 754**
Zeros messages	**4 384**
Messages for the statistics	**206 370**

Table 28. Number of tested messages

The 'BDS_number_guess' variable characterizes how many different BDS numbers were assigned to one message. In case of the received ADS-B data were allocated at most three BDS numbers to one row. Mostly, a message had only one assignment, specifically more than eighty percent of the non-zero messages were identified unambiguously. Columns where one row was padded with non-zero values more than once occurred only rarely. Two assignments represent approximately 3% of the cases and even three different allocations of the BDS number were found out. Fortunately, the heuristic algorithm is able to solve these conflicts and in every case a final BDS number or the 'Unknown' symbol is assigned to the message based on specific conditions. The layout of numbers of direct or ambiguous assignment is characterized in Table 29.

Category	Number	Percentage*
No assignment (without zero messages)	32 329	15,67%
One assigned BDS number	167 790	81,31%
Two assigned BDS numbers	6 224	3,02%
Three assigned BDS numbers	27	0,01%

Table 29. Overview of BDS numbers assignment

*** the percentage was calculated without including zeros messages**

Thanks to the heuristics algorithm, almost 85% of the tested messages were identified one of ELS, EHS or MRAR BDS registers. In contrast, the program is not able to recognize more than 15% of received replies to SSR interrogation. These messages may contain different BDS numbers which are not included in the algorithm testing, as well as erroneously received messages. The CRC check is not sufficient to exclude saving of all incorrectly collected data. Some of the decoded 24-bit addresses may not correspond any interrogated aircraft. A redundant bit shift may occur within the received data without revealing the error. Thus, the sample of data contain various rows with messages which are not distinguished by the algorithm. The result of the final BDS number assignment are specified in Table 30.

Category	Number	Percentage*
Identified messages	174041	**84,33%**
Unknown messages	32329	**15,67%**

Table 30. Percentage degradation of the messages identification

*** the percentage was calculated without including zeros messages**

In case of EHS BDS registers, they represent a majority of all received replies on the SSR Mode S interrogation. They create together approximately three quarters of tested messages. In other words, each of EHS BDS numbers are identified in more than 20% of examined rows. Any of ELS BDS registers does not exceed a percentage of 5%. Actually, one message was determined BDS register $3{,}0_{16}$, which is considered an error. It is estimated that the 'ACAS Resolution Advisory' is spontaneously transmitted in the order of frequency three

times a day. This register can be received providing that a conflict situation has occurred. Then, more than one message shall contain the BDS code $3,0_{16}$. As it was assumed, MRAR BDS registers rarely occurred. Specifically, only fifteen messages were identified the BDS register $4,4_{16}$. Currently, the heuristic algorithm does not assign the BDS $4,5_{16}$ number, so it is considered correct not to distinguish this register. A distribution within the 84,33% of BDS registers with assigned a specific BDS number is characterized in Table 31 below.

Category	Number	Percentage
BDS code $1,0_{16}$	3848	1,86%
BDS code $2,0_{16}$	7099	3,44%
BDS code $3,0_{16}$	1	0,00%
BDS code $4,0_{16}$	52808	25,59%
BDS code $5,0_{16}$	41888	20,30%
BDS code $6,0_{16}$	60077	29,11%
BDS code $4,4_{16}$	15	0,01%
BDS code $4,5_{16}$	0	0,00%
BDS code $1,7_{16}$	8305	4,02%

Table 31. Distribution of particular BDS codes of identified messages

In this case, different samples of data from ADS-B receivers have been examined, their statistics have reported similar allocation of BDS number assignments as well as other tested indicators.

5 Success Rate Evaluation

However, the sample of data from ADS-B receivers proves that the heuristic algorithm is able to assign a BDS number to a tested message, this information is not sufficient to evaluate whether the BDS number assignment is correct or not. For this purpose, a sample of data from real operation provided by ATN CR was tested by the algorithm. This data contains known BDS register numbers corresponding to binary sequence of 56-bit bits of BDS registers which enables comparison the known number to a number assigned by the algorithm for the same message. According to this comparison, a success rate of the BDS number assignment may be evaluated for the created heuristic algorithm.

5.1 ATN CR Data Structure

The decoded ASTERIX data provided by ATN CR was received within ten minutes to midnight on October 10 this year. An original '.xml' format was converted into an appropriate MAT-file by the 'Initial_data_processing_ATN.m' script, as well as the hexadecimal message into binary code. The MAT-file consists of two variables, the first one containing 56-bits binary sequences, and the second one where a known BDS number is saved. The originally provided BDS numbers correspond to numbers of rows of the 'BDSmessages' variable with the binary BDS codes.

The sample of data chosen for the success rate evaluation of the heuristic algorithm was called the 'ThirdAsterixBDS.mat'. There were more samples of data obtained from ATN CR. The chosen data contains the most of messages for testing and is the most up-to-date sample. A total number of examined rows is more than thirty-eight thousand which was considered appropriate for the evaluation.

The sample of data is naturally consisted of BDS registers of ELS and EHS. MRAR is represented by an occurrence of BDS register $4{,}4_{16}$. Nevertheless, some BDS codes which are not classified as one of these groups occur within the sample of data. The heuristic algorithm assigns the 'Unknown' symbol to these registers. The overview of the ATN data structure is specified in Table 32.

Category	Number	Percentage
Total ELS, EHS, MRAR	37610	98,12%
Total Unknown	721	1,88%
Total	38331	100,00%

Table 32. Distribution of ATN data

EHS registers cover the main part of the data, as well as in case of the data received by ADS-B receivers. The 'Heading and speed report' is regarded the most transmitted

BDS register of all examined ones. Almost 40% of the messages within the data sample was marked as the BDS code $6,0_{16}$. The BDS register $4,0_{16}$ achieved only slightly less frequency of receiving and the 'Track and turn report', the last one of the EHS group, was contained in more than 20% rows. In other words, EHS creates more than 95% of all BDS registers decoded from the ASTERIX data. Not every ELS BDS code type is represented within the data sample. Both, the 'Data link capability report' and the 'Common usage GIBC capability report' occurs in less than 1% of cases. The BDS register $3,0_{16}$ is decoded only on condition that the RA is initiated by ACAS system which is unlikely to record within the data. Therefore, there is no occurrence of this BDS code. Conversely, the 'Aircraft identification' shall be sent by the Mode S transponder by default. However, an information from the BDS register $2,0_{16}$ is directly decoded into a 'TId' component of the processed data by the Croatia Control program and it is not saved into a 'MBdata' like by other registers, see Figure 33.

```
<TId>AFL2327</TId>
<MBdata>91CA0F30BFFC00</MBdata>
<DSB1>6</DSB1>
<DSB2>0</DSB2>
```

Figure 33. BDS code $2,0_{16}$ location within ATN data

The 'Meteorological routine air report' represents the last from tested BDS numbers. As aforementioned, BDS code $4,5_{16}$ is not transmitted. Thus, MRAR BDS registers are specified by less than 1% of all the data. The significant part of the ATN data is created by undefined BDS registers. These BDS numbers are not examined within the created heuristic algorithm neither are considered ELS, EHS or MRAR registers. According to Manual on Mode S Specific Services [5], a BDS code $E,1_{16}$ and $E,2_{16}$ contains 'Reserved for Mode S BITE' (Built-in Test Equipment). A BDS code $1,D_{16}$ is included in one of the 'Mode S Specific Services Capability Report' as well as all BDS registers between numbers $1,8_{16}$ and $1,F_{16}$. A BDS register $0,0_{16}$ is undefined and shall not be decoded. This BDS number assignment is considered erroneous and caused some issues connected with its identification which is explained in the following chapter. Representation of particular BDS registers decoded from the ASTERIX data is characterized in Table 33.

BDS code Number	BDS code Name	Number	Percentage
BDS code $1,0_{16}$	Data link capability report	50	0,13%
BDS code $4,0_{16}$	Selected vertical intention	14046	36,64%
BDS code $5,0_{16}$	Track and turn report	7843	20,46%
BDS code $6,0_{16}$	Heading and speed report	15124	39,46%
BDS code $4,4_{16}$	Meteorological routine air report	239	0,62%
BDS code $1,7_{16}$	Common usage GIBC capability report	308	0,80%
BDS code $E,1_{16}$	Reserved for Mode S BITE	443	1,16%
BDS code $E,2_{16}$	Reserved for Mode S BITE	38	0,10%
BDS code $1,D_{16}$	Mode S Specific Services Capability Report	73	0,19%
BDS code $0,0_{16}$	Undefined	167	0,44%
Total		38331	100,00%

Table 33. Data structure of particular BDS registers

Equally to the ADS-B data sample, also here may be found rows which are padded only with zeros. In this case, zeros messages represent more than 1% of all data. Almost every zero message is included into not examined BDS register. Only a few messages were marked as the BDS number $4,0_{16}$. The percentage within the whole sample as well as within these BDS registers is negligible, see Table 34.

Category	Number	Percentage of all messages	Percentage within the category
Zeros messages	432	1,13%	100,00%
BDS code $4,0_{16}$	11	0,03%	0,08%
Unknown	421	1,10%	58,39%

Table 34. Number of zero messages

Zero messages are excluded in the beginning of the algorithm testing for the BDS number assignment, and the number of zero-bits sequences within a group of BDS codes $4,0_{16}$ is considered insignificant. Therefore, the success rate evaluation involved also these rows and is calculated from the total number of messages.

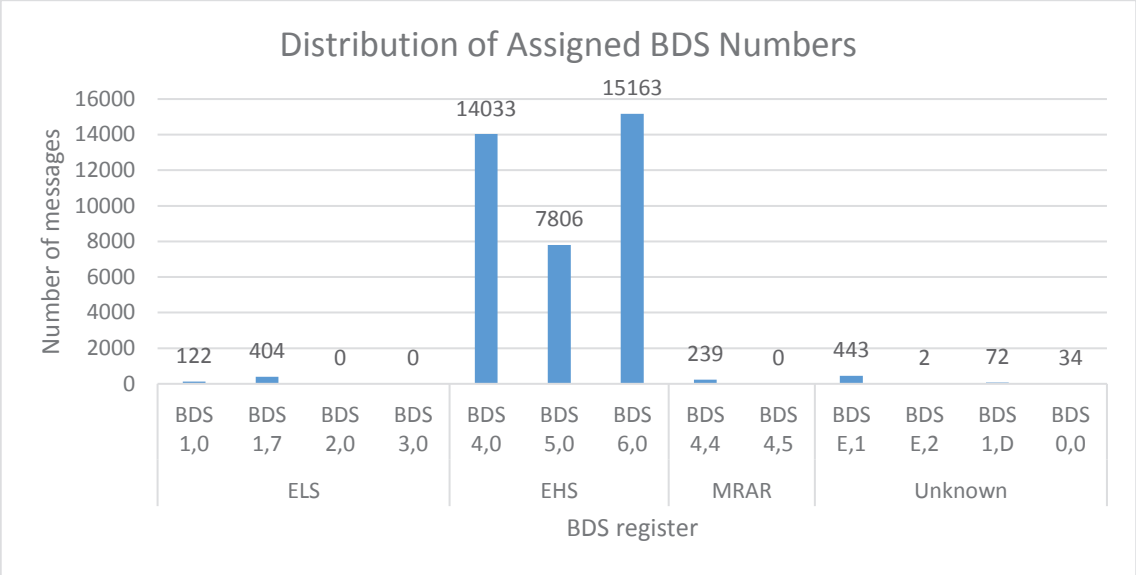
This sample of data may evaluate the success rate of the BDS number assignment for six BDS registers as well as for the unknown ones.

5.2 Algorithm Evaluation

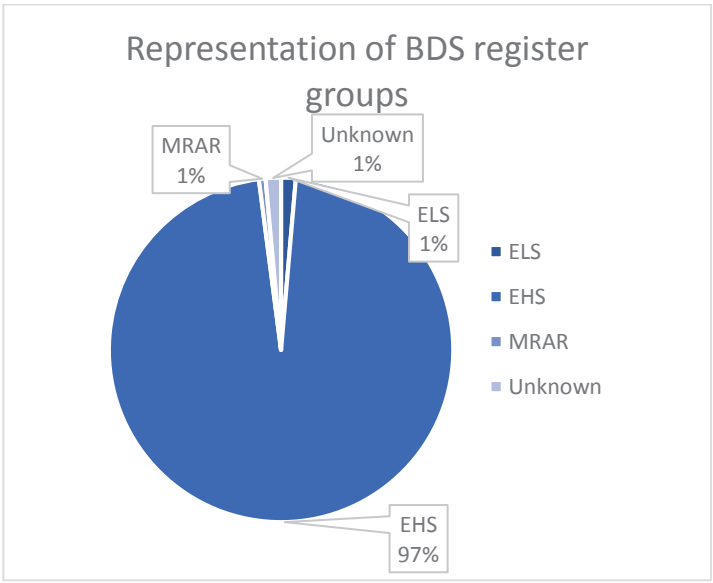
The MAT-file 'ThirdAsterixBDS.mat' is chosen as an evaluating input for the created heuristic algorithm. Every assigned BDS number is compared to the known number from the decoded sample of data provided by ATN CR.

The algorithm assigned a particular BDS number to every message or the message was market as 'Unknown'. The data structure according to the distinguishing algorithm is

characterized by number of messages allocated to one BDS number and representation of BDS register groups in Graphs 1 and 2.



Graph 1. Distribution of assigned BDS numbers to ATN data



Graph 2. Representation of allocated BDS groups within ATN data

On condition that the BDS number identified by the heuristic algorithm corresponds to the known BDS number from the ASTERIX data, this BDS number is regarded as a correct assigned. Conversely, some wrong BDS number allocations may occur by distinguishing by the algorithm. The wrong assignment is considered a different BDS number assigned to the known BDS register by the program. In other words, the BDS register is incorrectly identified or marked as unknown. A number of erroneous allocated BDS number is calculated as difference between a number of the BDS register assignment and a number of known

the same BDS number within the ATN data enlarged of the difference between a number of the known BDS number and a number of the correct assigned BDS number. For this reason, the number of wrong assignments may exceed the original number of the BDS number allocations. Then, the success rate is simply calculated according to a formula (5.1). Similarly, an error rate can be evaluated as in a formula (5.2).

$$\text{Success Rate} = \left(\frac{\text{Correct Assigned BDS Number}}{\text{ASTERIX BDS Number}}\right) \cdot 100\% \quad (5.1)$$

$$\text{Error Rate} = \left(\frac{\text{Wrong Assigned BDS Number}}{\text{ASTERIX BDS Number}}\right) \cdot 100\% \quad (5.2)$$

Both rates were evaluated for the six BDS numbers which were possible to compare, as well as for unknown BDS registers. With exception of the unknown BDS numbers, the success rate overcomes 95% on a case by case basis. The total success rate of the BDS number assignment by the heuristic algorithm is equal to 99,42%. In less than two percent of the cases, the BDS number may be assign incorrectly. A sum of both rates may exceed 100%, as well as the error rate, which is caused by the calculation of wrong assigned BDS codes. For example, two messages were not assigned to BDS code $1{,}0_{16}$, even if they were originally considered this BDS register number, and seventy-two rows were assigned this BDS code, even if they were not considered this BDS number according to the ASTERIX data. The number of the known 'Data link capability report' is equal to fifty, so both calculations overcome 100%. The success rate and the error rate, as well as numbers of correctly and erroneously identified BDS numbers are specified in Table 35.

BDS code number	Algorithm Assigned	ASTERIX BDS	Correct Assigned	Wrong Assigned	Success rate	Error rate
BDS code $1{,}0_{16}$	122	50	48	74	96,00%	148,00%
BDS code $2{,}0_{16}$	0	0	0	0	-	-
BDS code $3{,}0_{16}$	0	0	0	0	-	-
BDS code $4{,}0_{16}$	14033	14046	14033	13	99,91%	0,09%
BDS code $5{,}0_{16}$	7806	7843	7805	75	99,52%	0,96%
BDS code $6{,}0_{16}$	15163	15124	15124	39	100,00%	0,26%
BDS code $4{,}4_{16}$	239	239	239	0	100,00%	0,00%
BDS code $4{,}5_{16}$	0	0	0	0	-	-
BDS code $1{,}7_{16}$	404	308	308	96	100,00%	31,17%
Unknown codes	564	721	551	327	76,42%	45,35%
Total	38331	38331	38108	624	**99,42%**	**1,63%**

Table 35. The algorithm success and error rate

The BDS register $6{,}0_{16}$, the BDS register $4{,}4_{16}$ and the BDS register $1{,}7_{16}$ were identified whenever they occurred. The ideal assignment is represented by the 'Meteorological routine air report', because there was not even any error by its distinguishing. High success rate and low erroneous rate are specific for all EHS BDS registers. As aforementioned, the identification

of BDS code $1,7_{16}$ is considered difficult because of insufficient significant parts in its structure. Almost one third of the assigned BDS number $1,7_{16}$ by the algorithm were unknown or the undefined BDS register according to the ASTERIX data. Due to that fact, the 'Common usage GIBC capability report' shall be recognised in every case, but one third of this BDS number assignment may represent a different BDS register. The structure of some messages of the undefined BDS code $0,0_{16}$ were identical to the bit distribution contained in the 'Data link capability report'. Therefore, the error rate of the BDS register $1,0_{16}$ achieved so high number. It may not be an algorithm fault, necessary. The undefined BDS code may include some of the real defined 56-bits sequences, as for example BDS code $1,0_{16}$. More important fact is, that only two messages were not identified like this BDS register and its success rate exceeded ninety-five percent. The unknown BDS codes create the least successful part of the algorithm identifying. They cause the incorrect BDS number assignment which is characterized by large amount of allocated BDS number by the algorithm than the number of the ASTERIX known number.

The most problematic unknown BDS register is considered $E,2_{16}$. It was identified as BDS register $1,7_{16}$ in most of its occurrence. The undefined BDS code causes errors not only by distinguishing the 'Common usage GIBC capability report', but also the BDS register $1,0_{16}$. Conversely, BDS codes $E,2_{16}$ and $1,D_{16}$ were generally marked as 'Unknown'. The particular cause of the erroneous assignment is described in Table 36 below.

BDS code number	Identified as 'Unknown'	ASTERIX BDS	Correct Assigned	Wrong Assigned	Success rate	Error rate
BDS code $E,1_{16}$	443	443	443	0	100,00%	0,00%
BDS code $E,2_{16}$	2	38	2	36	5,26%	94,74%
BDS code $1,D_{16}$	72	73	72	1	98,63%	1,37%
Undefined BDS code	34	167	34	133	20,36%	79,64%

Table 36. Rates of unknown BDS registers

Hopefully, the remaining BDS registers $2,0_{16}$, $3,0_{16}$ and $4,5_{16}$ achieves similar results of the success rate as the evaluated BDS codes.

6 Meteorological Decoder

A MATLAB script 'Meteo_Decoder.m' has been created for obtaining meteorological information from on-board of an airplane, which is contained in the 'Meteorological routine air report' and the 'Meteorological hazard report'. The data is decoded according to the Technical Provisions for Mode S Services and Extended Squitter [5] and recommendations obtained from ATN CR.

To demonstrate a functionality of the decoder, an input for the decoding program is based on the previous identification of BDS registers and is characterised by a MAT-file 'AsterixMeteoBDS.mat'. For different purposes, there can be used any messages in 56-bits binary sequences. The input file shall contain messages in binary code of the BDS register, which was assigned the number $4{,}4_{16}$ or $4{,}5_{16}$ to. Unfortunately, there is no occurrence of the BDS code $4{,}5_{16}$ within the provided data and currently there is no interrogation of this register.

Particular fields of the message specifying meteorological information are converted into the decimal code with help of a 'DecodingMeteo.m' function and further recalculated when it is needed to gain the value in a corresponding unit. Meteorological values cannot be only expressed with numbers, but also some of them are characterised by a word interpretation appropriate to a phenomenon strength. On the purpose to assign the word expression, a function 'ValueInterpretation.m' has been created. After the decoding process, the meteorological data are saved into variables and into a MAT-file 'DecodedMeteo.mat'. The sample of decoded data is characterised in Figure 34 below.

DecodedBDSFourFour{2,1} <1x7 cell>

	1	2	3	4	5	6	7
1	'INS'	8	240.4688	-16	0	0	0

Figure 34. Sample of decoded meteorological data from BDS $4{,}5_{16}$

Decoding of particular registers is described in following chapters. The whole script for decoding of meteorological messages is specified in Appendix E and the completing function for the value interpretation in Appendix F.

6.1 Decoding of the BDS register $4{,}4_{16}$

There are seven particular data fields contained in the 'Meteorological routine air report', which are characterised by their first and the last bit position, a number of bits, a range, a LSB, a MSB and units. Every field with an exception of 'FOM/SOURCE' data is specified by a status bit, which setting determines whether the following field provides meteorological information

or not. The meteorological decoder is created on base of the layout symbolized in Table 37 below.

Field number	Bit position	Number of bits	Data set	Range	LSB	MSB	Unit
1	1-4	4	FOM/SOURCE	1 - 16	-	-	-
2	5	1	Status	-	-	-	-
	6-14	9	Wind speed	0 - 512	1	256	[knots]
3	15-23	9	Wind direction	0 - 360	180/256	180	[degrees]
4	24	1	Status	-	-	-	-
	25	1	Sign	-	-	-	-
	25-34	10	Static air temperature	-128 - +128	0,25	64	°C
5	35	1	Status	-	-	-	-
	36-46	11	Average static pressure	0 - 2048	1	1024	[hPa]
6	47	1	Status	-	-	-	-
	48-49	2	Turbulence*	-	-	-	-
7	50	1	Status	-	-	-	-
	51-56	7	Humidity	0 - 128	64/100	64	[%]

Table 37. BDS register $4,4_{16}$ decoding information

*** turbulence interpretation of two bits assigned turbulence, see Table A [5]**

The number of messages for decoding is saved into a 'm' variable, which is further used for creating a for-loop. Significant bits are represented by vectors with a bit position which the field starts with and ends with. A variable 'DecodedBDSFourFour' is prepared for a storage of decoded data. This variable is created in the cell format to enable saving non-numeric values, as obvious in Figure 35.

```
% BDS 4,4 decoding
m=size(BDSFourFour,1); %number of rows

%significant bits for BDS 4,4
FourFourStartBit=[1 6 15 25 36 48 51];
FourFourEndBit=[4 14 23 34 46 49 56];
DecodedBDSFourFour=cell(m,1);
```

Figure 35. Defining variables for BDS register $4,4_{16}$ decoding

The for-loop has been created to enable decoding all meteorological information from the first message one by one. The tested message is saved into a variable and all fields characterized by significant bits are converted into the decimal scale by the simple function 'DecodingMeteo.m'. The tested message, the position of start of the field and its end symbolise inputs for the function. Particular fields are decoded with help of the MATLAB function 'bi2de' and their decimal values are saved into a 'Decode' variable which contains the output of the function, as characterized in Figure 36.

```
function [Decode] = DecodingMeteo(Message,FieldStart,FieldEnd)

Decode=[]; %zero decoding

for a=1:size(FieldStart,2); %all status fields
    Decode(1,a)=bi2de(Message(1,FieldStart(1,a):FieldEnd(1,a)),'left-msb');
end

end
```

Figure 36. 'DecodingMeteo.m' Function

Some of the fields converted to the decimal scale may be considered fully decoded and they do not need any other decoding steps. This is for meteorological information about the wind speed and the average static pressure. On the other hand, remaining fields shall be further recalculated to obtain the required value. A field containing the wind direction has to be divided by 256 and multiplied by 180 to get correct degrees of the direction. A sign of the BDS code $4,5_{16}$ is characterized by a bit on position 25. This bit decides about a positive or negative value of the measured temperature. If it is set at '0', the value shall be positive and only a multiplication by 0.25 is necessary to achieve the appropriate value in the degree Celsius unit. In the opposite case, the sign value, which is equal to 1024 in the decimal scale, is deducted from the number in the temperature field before the identical multiplication with positive value. To ensure percentage of humidity, the last decoded field has to be multiplied by a fraction 64/100. Corresponding recalculations in the created script are presented in Figure 37.

```
%Recalculation of wind direction
DecodedFourFour(1,3)=DecodedFourFour(1,3).*(180/256);

%Recalculation of temperature
if TestedMessage(1,25)==0; %positive values
      DecodedFourFour(1,4)=(DecodedFourFour(1,4).*0.25);
elseif TestedMessage(1,25)==1; %negative values
      DecodedFourFour(1,4)=((DecodedFourFour(1,4)-1024).*0.25);
end

%Recalculation of humidity
DecodedFourFour(1,7)=DecodedFourFour(1,7).*(64/100);
```

Figure 37. Recalculation of fields in BDS code $4,4_{16}$

An information about the FOM/SOURCE parameter is coded in the beginning of the message; in its first four bits. Corresponding bits are converted into the decimal system and decoded according to the Table 38.

Decimal code	FOM/SOURCE
0	Invalid
1	INS
2	GNSS
3	DME/DME
4	VOR/DME
5-15	Reserved

Table 38. FOM/SOURCE data interpretation

Decoded fields are converted from the double format to the cell format before the FOM/SOURCE field decoding. This step has to be done, because the sources are expressed by words. Subsequently, one interpretation is assigned to a corresponding decimal value. However, values 'Invalid' and 'Reserved' should not occur due to their exclusion within identifying, see Figure 38.

```
%FOM/SOURCE decoding
FOM_SOURCEoptions={'Invalid','INS','GNSS','DME/DME','VOR/DME','Reserved'};

if DecodedFourFour{1,1}==0;
    DecodedFourFour{1,1}=FOM_SOURCEoptions(1,1);
elseif DecodedFourFour{1,1}==1;
    DecodedFourFour{1,1}=FOM_SOURCEoptions(1,2);
elseif DecodedFourFour{1,1}==2;
    DecodedFourFour{1,1}=FOM_SOURCEoptions(1,3);
elseif DecodedFourFour{1,1}==3;
    DecodedFourFour{1,1}=FOM_SOURCEoptions(1,4);
elseif DecodedFourFour{1,1}==4;
    DecodedFourFour{1,1}=FOM_SOURCEoptions(1,5);
else DecodedFourFour{1,1}=FOM_SOURCEoptions(1,6);
end
```

Figure 38. FOM/SOURCE field decoding

The strength of the turbulence is expressed by a word value, as well as any hazardous phenomena. The function 'ValueInterpretation' has been created for these phenomena. Providing that status bit is set at one, the assignment of bit setting and the word expression is made according to the Table 39 below.

Bit 1	Bit 2	Decimal value	Interpretation*
0	0	0	Nil
0	1	1	Light
1	0	2	Moderate
1	1	3	Severe

Table 39 - Interpretation of hazard data set

***Terms light, moderate and severe are defined in the PANS-ATM (Doc 4444) [10]**

Firstly, the status bit is tested by decoding of the turbulence field. Then an allocation of the interpretation is processed with help of the 'ValueInterpretation' function, see the Figure 39.

```
%TURBULENCE decoding
if TestedMessage(1,47)==0;
DecodedFourFour{1,5}=0;
else
DecodedFourFour{1,5}=ValueInterpretation(DecodedFourFour{1,5});
end

%all messages saving
DecodedBDSFourFour{i}=DecodedFourFour;
```

Figure 39. Turbulence decoding and final saving

Finally, decoded values including the FOM/SOURCE, the wind speed and direction, the static air temperature, the average static pressure, the turbulence and the humidity information are saved into the variable 'DecodedBDSFourFour'.

6.2 Decoding of the BDS register $4,5_{16}$

Eight data fields containing meteorological information are found in the 'Meteorological hazard report'. Apart from the hazardous phenomena, the same data fields like in BDS register $4,4_{16}$ with information about temperature and pressure may be obtained from this register; as specified in Table 40. Last six bits are considered reserved, they shall be set at zeros and are not included into the decoding script.

Field number	Bit position	Number of bits	Data set	Range	LSB	MSB	Units
1	1	1	Status	-	-	-	-
	2-3	2	Turbulence	-	-	-	-
2	4	1	Status	-	-	-	-
	5-6	2	Wind shear	-	-	-	-
3	7	1	Status	-	-	-	-
	8-9	2	Microburst	-	-	-	-
4	10	1	Status	-	-	-	-
	11-12	2	Icing	-	-	-	-
5	13	1	Status	-	-	-	-
	14-15	2	Wake vortex	-	-	-	-
6	16	1	Status	-	-	-	-
	17	1	Sign	-	-	-	-
	18-26	10	Static air temperature	-128 - +128	0,125	64	°C
7	27	1	Status	-	-	-	-
	28-38	11	Average static pressure	0 - 2047	1	1024	[hPa]
8	39	1	Status	-	-	-	-
	40-51	12	Radio height	0 - 65528	16	32768	[ft]
-	52-56	6	Reserved	-	-	-	-

Table 40. BDS register $4,5_{16}$ decoding information [5]

Similarly to the BDS $4,4_{16}$, a number of identified messages of the 'Meteorological hazard report' is characterized in a variable 'n' which is used in following for-loops. The final decoded meteorological information is stored in a variable 'DecodedBDSFourFive' in the cell format which is defined in the beginning for decoding the BDS code $4,5_{16}$. The tested message is saved and its particular fields are decoded with help of 'DecodingMeto.m' function. A field contained the temperature information is divided in positive and negative values, as well as in case of the BDS register $4,4_{16}$. This decoding differs by the position of the sign bit which is equal to 17 here.

A field containing the radio height needs to be multiplied by 16 to ensure the required value in feet, otherwise there is no recalculation in BDS register $4,5_{16}$ decoding. Then, the decimal and recalculated values are converted into the cell format, see the Figure 40 below.

```
%Recalculation of radio height
DecodedFourFive(1,8)=DecodedFourFive(1,8)*16;

%Convertion to cell
DecodedFourFive=num2cell(DecodedFourFive);
```

Figure 40. Radio height recalculation and converting to cell

Hazardous phenomena are characterised by a pair of bits and their strength is expressed alike the turbulence information in the BDS register $4,4_{16}$. Therefore, meteorological data of the turbulence, the wind shear, the microburst, the icing and the wake vortex are decoded in the same way. An example is characterized in Figure 41 below.

```
%WIND_SHEAR decoding
if TestedMessage(1,4)==0;
DecodedFourFive{1,2}=0;
else
DecodedFourFive{1,2}=ValueInterpretation(DecodedFourFive{1,2});
end

%MICROBURST decoding
if TestedMessage(1,7)==0;
DecodedFourFive{1,3}=0;
else
DecodedFourFive{1,3}=ValueInterpretation(DecodedFourFive{1,3});
end

%ICING decoding
if TestedMessage(1,10)==0;
DecodedFourFive{1,4}=0;
else
DecodedFourFive{1,4}=ValueInterpretation(DecodedFourFive{1,4});
end
```

Figure 41. Sample of hazardous fields decoding

The decoded fields of the BDS register $4,5_{16}$ represented by information about the hazardous phenomena strength and numeric values of the static air temperature, the average static pressure and the radio height are saved into the variable 'DecodedBDSFourFive'.

In addition, both decoded registers may be saved into a MAT-file, as exemplified in Figure 42 below.

```
name=('DecodedMeteo');
file=['.\Decoded\',name,'.mat'];
save(file,'DecodedBDSFourFour','DecodedBDSFourFour');
```

Figure 42. An example of possible saving decoded values

7 Conclusion

In conclusion, the main objectives of this thesis have been successfully processed. Data samples received by ADS-B receivers as well as data proved by ATN CR were collected, filtrated and converted into the required form suitable for further testing. Significant parts of the messages were found thanks to the detailed bit analysis of ELS, EHS and MRAR BDS registers contained in replies on the SSR Mode S interrogation. This analysis together with other criteria enabled to create the heuristic algorithm for distinguishing particular BDS registers. The algorithm is able to assign the BDS number to an appropriate message for nine BDS registers transmitted by default. The functionality check of the algorithm was tested by the sample of data from the Laboratory of ATM Systems and the success rate was evaluated by comparison the algorithm results with the ASTERIX data from the real operation. The meteorological information was decoded from the BDS register $4{,}4_{16}$ and the decoding program may provide also decoded data about meteorological phenomena contained in the 'Meteorological Hazard Report'.

The algorithm assigned one of the examined BDS numbers to almost 85% of the passively received messages by the receivers. The unassigned percentage from the data sample may contain BDS registers which are not examined or incorrectly received messages with a noise. The success rate of the BDS number correct assignment was calculated more than 99%. Conversely, the wrong BDS number may be assigned in less than 2% of the cases. According to the expectations, the most part of the identified registers was represented by BDS registers of the Enhanced Surveillance, followed by the messages of the Elementary Surveillance and the least occurring registers were the meteorological BDS codes. The probability of the correct BDS number assignment may be increased with further analysis of the remaining registers which are not in the interests of this thesis, although they are certainly transmitted within the SSR Mode Addressed surveillance.

The particular outcomes are represented by MATLAB scripts 'Data_initial_processing_ADSB', 'Data_initial_processing_ATN', 'Heuristic_algorithm.m', 'Meteo_decoder.m', their associated functions and their outputs in the form of MAT-files. All programs codes are specified in appropriate Appendices. Together with the programming part, the description of the codes and the data analysis are considered desirable to complete the outcome.

The thesis output provides sources for the next research activities for the Laboratory of ATM Systems at the CTU in Prague. On the basis of identified BDS registers numbers, corresponding decoded programs may be created for obtaining different flight information. The decoded data is considered crucial for further analysis, statistics or databases. In addition, the MRAR information are useful for obtaining the meteorological data from

the high altitude without the difficult measurement by meteorological probes. It can be used as an additional data for an up-to-date investigation dealing with a contrails formation. While the BDS register $4,5_{16}$ is transmitted, various information may be obtain about hazardous meteorological phenomena.

References

[1] ICAO, *Doc. 9684: Manual of the Secondary Surveillance Radar (SSR) Systems, AN/951* .3rd ed. 2004, ISBN 92-9194-333-9.

[2] ICAO, *Annex 10 Aeronautical Telecommunication, Volume IV.*

[3] Frei J. *Moznosti ziskani a vyuziti meteo dat z BDS registru pro potreby uzivatelu vzdusneho prostoru*. Prague, 2016. Doctoral thesis. CTU in Prague.

[4] ICAO, *Annex 10 Aeronautical Telecommunication, Volume III*

[5] ICAO, Doc 9871: *Technical Provisions for Mode S Services and Extended Squitter, AN/464.* 1st ed. 2008, ISBN 978-92-9231-117-9.

[6] Grappel, R. D., Harris, G. S., Kozar, M. J., Wiken, R. T. *Elementary Surveillance (ELS) and Enhanced Surveillance (EHS) Validation via Mode S Secondary Radar Surveillance*, Project report ATC-337, ESC-TR- 3007-054, Lincoln Laboratory, Massachusetts Institute of Technology, 2008.

[7] *Aircraft Equipage Requirements in the European Commission IRs 1207/2011 and 1028/2014* [online]. EUROCONTROL [Accessed 2017-03-19]. Available at http://www.eurocontrol.int/spi-ir

[8] Haan, de S., Sondij, J., Philippi, H., Strajnar, B., Hoff., A., Besson, F., Vedel, H., Garcia-Moya., J. a., Stone, E., Riskila E. *Final report of the EUMETNET ADDD FS ET on the feasibility to initiate a new EUMETNET activity for aircraft derived observations under the EUMTENET observation program*, 2. 10. 2015

[9] Zach, Martin. *Navrh nizkonakladoveho MLAT systemu*. Prague, 2015. Master's thesis. CTU in Prague. Supervisor Ing. Stanislav Pleninger, Ph.D.

[10] ICAO, Doc 4444: Procedures for Air Navigation Services - *Air Traffic Management.* 16th ed. 2016, ISBN 978-92-9258-081-0

[11] EPA, *Aircraft Contrails Factsheet* [online]. [Accessed 2017-03-25]. Available at https://www.faa.gov/regulations_policies/policy_guidance/envir_policy/media/contrails.pdf

[12] ICAO, Doc 9871: *Technical Provisions for Mode S Services and Extended Squitter, AN/464*. 2nd ed. 2011, ISBN 978-92-9231-117-9.

[13] ICAO, Doc 9866: *Manual on Mode S Specific Services, AN/952.* 2nd ed. 2004, ISBN 92-9194-407-6

[14] *Aircraft meteorological data relay (AMDAR) reference manual*. Geneva, Switzerland: Secretariat of the World Meteorological Organization, 2003. WMO (Series), no. 958. ISBN 92-63-10958-3.

[15] Commission Implementing Regulation (EU) No 1028/2014 [online]. Official Journal L 284/7. 2014 [Accessed 2017-03-22]. Available at http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv%3AOJ.L_.2014.284.01.0007.01.ENG

[16] Commission Implementing Regulation (EU) No 1207/2011 [online]. Official Journal L 305/35. 2011 [Accessed 2017-03-22]. Available at http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32011R1207.ENG

[17] Cormen, Thomas H. *Introduction to algorithms.* 3rd ed. Cambridge, Mass.: MIT Press, 2009. ISBN 978-0-262-03384-8.

[18] EUROCONTROL ACID Programme, ELS_OPS-TF members, *Mode S Elementary Surveillance (ELH) Operations Manual, 2010.*

[19] EUROCONTROL, *Mode S: Harmonisation of the Transition Arrangements for State Aircraft, Process and Procedures for the Managing of the Mode S Compliance Status and the Exemptions Granted to State Aircraft*, 14.1.2015

[20] EUROCONTROL Specification for Surveillance Data Exchange - Part 1: All Purpose Structured EUROCONTROL Surveillance Information Exchange (ASTERIX). *: EUROCONTROL-SPEC-0149* [online]. 2016 [Accessed 2017-5-24]. Availiable at http://www.eurocontrol.int/sites/default/files/content/documents/nm/asterix/Part%201%20-%20EUROCONTROL%20Specification%20ASTERIX%20%28SPEC-149%29%20Ed%202.4.pdf

[21] EUROCONTROL, *Target Reports Standard Document for Surveillance Data Exchange: Transmission of Monoradar, Part 4*, 2000.

[22] Grappel, R. D., Wiken, R. T. *Guidance Material for Mode S-Specific Protocol Application Avionics*, Project report ATC-334, ESC-TR-2006-082, Lincoln Laboratory, Massachusetts Institute of Technology, 2007.

[23] Haan, de S., Sondij, J., Philippi, H., Strajnar, B., Hoff., A., Besson, F., Vedel, H., Garcia-Moya., J. a., Stone, E., Riskila E. *Final report of the EUMETNET ADDD FS ET on the feasibility to initiate a new EUMETNET activity for aircraft derived observations under the EUMTENET observation program*, 2. 10. 2015

[24] Haan, de S. *Quality assessment of high resolution wind and temperature observation from Mode S*, Scientific Report No. WR 2009-07, De Bilt 2010.

[25] Haan, de S. *An improved correction method for high quality wind and temperature observations derived from Mode-S EHS*, Technical report TR-338, De Bilt 13. 6. 2013.

[26] Haan, de S., Haij, de M., Sondij. J. *The use of a commercial ADS-B receiver to derive upper air wind and temperature observations from Mode-S EHS information in the Netherlands*, Technical report TR-336, De Bilt, 2013.

[27] Haan, de S., Bailey, L. J., Konnen, J. E. *Quality assessment of Automatic Dependent Surveillance Contract (ADS-C) wind and temperature observation from commercial aircraft, Atmospheric MeasurementTechnics*, 2013, doi: 10.5194/amt-6-199-2013.

[28] Haan, de S., *Availibility and quality of Mode-S MRAR (BDS4.4) in MUAC area: a first study, KNMI internal report*, IR 2014-01, 7. 5. 2014.

[29] Haan, de S., Sondij, J., Philippi, H., Strajnar, B., Hoff., A., Besson, F., Vedel, H., Garcia-Moya., J. a., Stone, E., Riskila E. *Final report of the EUMETNET ADDD FS ET on the feasibility to initiate a new EUMETNET activity for aircraft derived observations under the EUMTENET observation program*, 2015.

[30] Haan, de S., Stoffelen, A. *Assimilation of High-Resolution Mode-S Wind and Temperature Observations in a Regional NWP model, Weather and Forecasting*, 2012, vol. 27, str. 918-937, doi: http://dx.doi.org/10.1175/WAF-D-11-00088.1

[31] ICAO, Regional Workshop on Aeronautical Surveillance, ASI WS WP/03, 2011.

[32] MathWorks, *The official home of MATLAB software*, Available at https://www.mathworks.com/products/matlab.html

[33] Pham, D.T., Karaboga D. *Intelligent optimisation techniques genetic algorithms, tabu search, simulated annealing and neural networks*. London: Springer, 2000. ISBN 9781447107217.

[34] Strajnar B. *Analysis and processing of Czech Mode-S observations*, 2015

[35] Strajnar, B., Zagar, N., Berre, L. *Impact of new aircraft observations Mode-S MRAR in a mesoscale NWP model, Journal of geophysical research*, vol. 120, issue 9, publicated on 16th of June 2015, doi: 10.1002/2014JD022654.

[36] Walter, R. *Flight Management Systems: Digital Avionics Handbook*. 2nd ed. CRC Press, 2007.

[37] Williamson, *S. Gill. Combinatorics for computer science*. Mineola, N.Y.: Dover Publications, 2002. ISBN 978-0-486-42076-9

List of Figures

Figure 1. Compatibility between SSR Mode A/C and Mode S [1] 11
Figure 2. Comm-B altitude reply, DF 20 13
Figure 3. Thesis workflow chart 17
Figure 4. SSR Mode S ELS coverage above Europe [8] 19
Figure 5. SSR Mode S EHS coverage above Europe [8] 19
Figure 6. SSR Mode S MRAR coverage above Europe [8] 20
Figure 7. Sample of data received by Radarcape receivers 21
Figure 8. Sample of ATN data 22
Figure 9. Defined variables for initial data processing 27
Figure 10. Filtration of long messages with DF=20 or DF=21 28
Figure 11. CRC and bit setting check 28
Figure 12. Filtration of required values of ATN data 30
Figure 13. Initial variables defining 37
Figure 14. Excluding of zeros messages 38
Figure 15. 'Decoding.m' function 39
Figure 16. BDS1 and BDS2 decoding and testing of BDS code $1,0_{16}$ 39
Figure 17. Sample of testing BDS code $2,0_{16}$ 40
Figure 18. Testing of BDS code $3,0_{16}$ 41
Figure 19. Condition testing for BDS code $1,7_{16}$ 41
Figure 20. Example of defining variables for BDS $4,0_{16}$ 42
Figure 21. BDS register $4,0_{16}$ testing 42
Figure 22. BDS register $5,0_{16}$ testing 44
Figure 23. BDS register $6,0_{16}$ testing 45
Figure 24. BDS register $4,4_{16}$ testing 45
Figure 25. BDS register $4,5_{16}$ testing 46
Figure 26. Direct BDS number assignment 48
Figure 27. An example of collision between two BDS numbers resolution 48
Figure 28. Solution of three BDS numbers assignment 49
Figure 29. Defining variables with appropriate size 49
Figure 30. Including messages to their corresponding groups 50
Figure 31. Saving identified BDS registers into MAT-file 50
Figure 32. The variable 'BDSmessages' 51
Figure 33. BDS code $2,0_{16}$ location within ATN data 55
Figure 34. Sample of decoded meteorological data from BDS $4,5_{16}$ 60
Figure 35. Defining variables for BDS register $4,4_{16}$ decoding 61
Figure 36. 'DecodingMeteo.m' Function 62
Figure 37. Recalculation of fields in BDS code $4,4_{16}$ 62
Figure 38. FOM/SOURCE field decoding 63
Figure 39. Turbulence decoding and final saving 64
Figure 40. Radio height recalculation and converting to cell 66
Figure 41. Sample of hazardous fields decoding 66
Figure 42. An example of possible saving decoded values 66

List of Tables

Table 1. ELS BDS registers [3] 14
Table 2. EHS BDS registers [3] 15
Table 3. EHS BDS registers provided data sets [5] 15
Table 4. MRAR BDS registers [3] 16
Table 5. MRAR BDS registers provided data sets [5] 16
Table 6. SSR operated in the Czech Republic 20
Table 7. Data fields specifications for BDS $4,4_{16}$ [5] 24
Table 8. BDS register $4,5_{16}$ decoding information [5] 25
Table 9. BDS code $1,0_{16}$ significant bits 31
Table 10. BDS code $1,7_{16}$ significant bits 32
Table 11. BDS1 and BDS2 setting for BDS code $2,0_{16}$ 32
Table 12. Value options for Aircraft identification 33
Table 13. BDS1 and BDS2 setting for BDS code $3,0_{16}$ 33
Table 14 - Significant bits setting for BDS code $3,0_{16}$ 34
Table 15. BDS code $4,0_{16}$ status bits 34
Table 16. BDS code $4,0_{16}$ reserved bits 34
Table 17. BDS code $5,0_{16}$ status bits 35
Table 18. BDS code $6,0_{16}$ status bits 35
Table 19. FOM/SOURCE data set decoding 35
Table 20. BDS code $4,4_{16}$ status bits 36
Table 21. BDS code $4,5_{16}$ status bits 36
Table 22. BDS code $4,5_{16}$ reserved bits 36
Table 23. BDS code $4,0_{16}$ values limitation 43
Table 24. BDS code $5,0_{16}$ values limitation 43
Table 25. BDS code $6,0_{16}$ values limitation 44
Table 26. BDS code $4,4_{16}$ values limitation 46
Table 27. BDS code $4,5_{16}$ values limitation 47
Table 28. Number of tested messages 51
Table 29. Overview of BDS numbers assignment 52
Table 30. Percentage degradation of the messages identification 52
Table 31. Distribution of particular BDS codes of identified messages 53
Table 32. Distribution of ATN data 54
Table 33. Data structure of particular BDS registers 56
Table 34. Number of zero messages 56
Table 35. The algorithm success and error rate 58
Table 36. Rates of unknown BDS registers 59
Table 37. BDS register $4,4_{16}$ decoding information [5] 61
Table 38. FOM/SOURCE data interpretation 63
Table 39 - Interpretation of hazard data set 63
Table 40. BDS register $4,5_{16}$ decoding information [5] 65

List of Graphs

Graph 1. Distribution of assigned BDS numbers to ATN data 57
Graph 2. Representation of allocated BDS groups within ATN data 57

List of Appendices

Appendix A. Bit Structure of ELS, EHS, MRAR BDS Codes 76
Appendix B. Initial Data Processing MATLAB Script for Data from ADS-B Receiver 85
Appendix C. Initial Data Processing MATLAB Script for ATN CR Data 87
Appendix D. Heuristic Algorithm MATLAB Script 89
Appendix E. Meteorological Decoder MATLAB Script 95
Appendix F. MATLAB Function for Hazardous Data Interpretation 97

Appendix A. Bit Structure of ELS, EHS, MRAR BDS Codes

1	MSB
2	
3	
4	BDS Code 1,0
5	
6	
7	
8	LSB
9	Continuation flag (see 9)
10	
11	
12	RESERVED
13	
14	
15	
16	Reserved for ACAS (see 1)
17	MSB
18	
19	
20	Mode S subnetwork version number (see 12)
21	
22	
23	LSB
24	Transponder enhanced protocol indicator (see 4)
25	Mode S specific services capability (see 2)
26	MSB
27	Uplink ELM average throughput capability (see 13)
28	LSB
29	Downlink ELM: throughput capability of downlink ELM
30	containing the maximum number of ELM segments that the
31	transponder can deliver in response to a single requesting
32	interrogation (UF = 24). (see 14)
33	Aircraft identification capability (see 11)
34	Squitter capability subfield (SCS) (see 5)
35	Surveillance identifier code (SIC) (see 6)
36	Common usage GICB capability report (see 7)
37	
38	RESERVED FOR ACAS (see 1)
39	
40	
41	MSB
42	
43	
44	
45	
46	
47	Bit array indicating the support status of DTE
48	Sub-addresses 0 to 1 5 (see 3 and 8)
49	
50	
51	
52	
53	
54	
55	
56	LSB

Figure 43. BDS code $1,0_{16}$ - Data link capability report [5]

1	0,5 Extended squitter airborne position
2	0,6 Extended squitter surface position
3	0,7 Extended squitter status
4	0,8 Extended squitter type and identification
5	0,9 Extended squitter airborne velocity information
6	0,A Extended squitter event-driven information
7	2,0 Aircraft identification
8	2,1 Aircraft registration number
9	4,0 Selected vertical intention
10	4,1 Next waypoint identifier
11	4,2 Next waypoint position
12	4,3 Next waypoint information
13	4,4 Meteorological routine report
14	4,5 Meteorological hazard report
15	4,8 VHF channel report
16	5,0 Track and turn report
17	5,1 Position coarse
18	5,2 Position fine
19	5,3 Air-referenced state vector
20	5,4 Waypoint 1
21	5,5 Waypoint 2
22	5,6 Waypoint 3
23	5,F Quasi-static parameter monitoring
24	6,0 Heading and speed report
25	Reserved for aircraft capability
26	Reserved for aircraft capability
27	E,1 Reserved for Mode S BITE (Built In Test Equipment)
28	E,2 Reserved for Mode S BITE (Built In Test Equipment)
29	F,1 Military applications
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	RESERVED
43	
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	

Figure 44. BDS code $1,7_{16}$ - Common usage GIBC capability report [5]

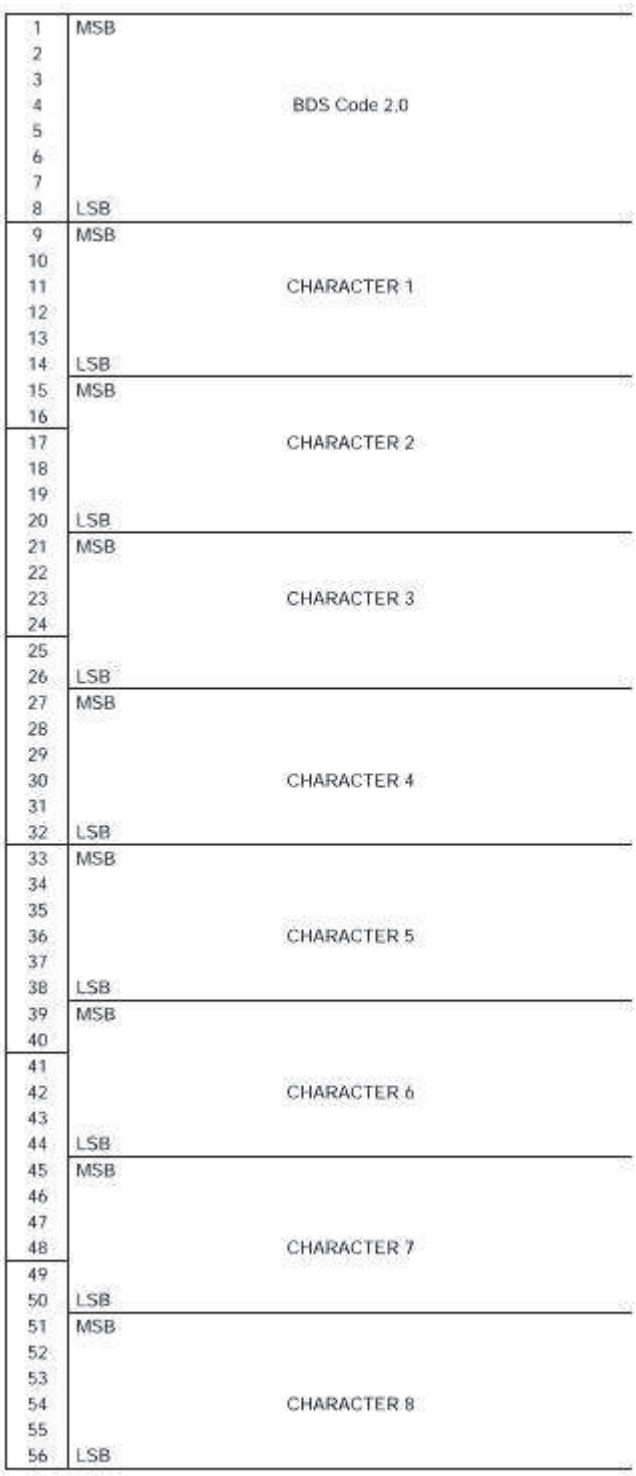


Figure 45. BDS code $2,0_{16}$ - Aircraft identification [5]

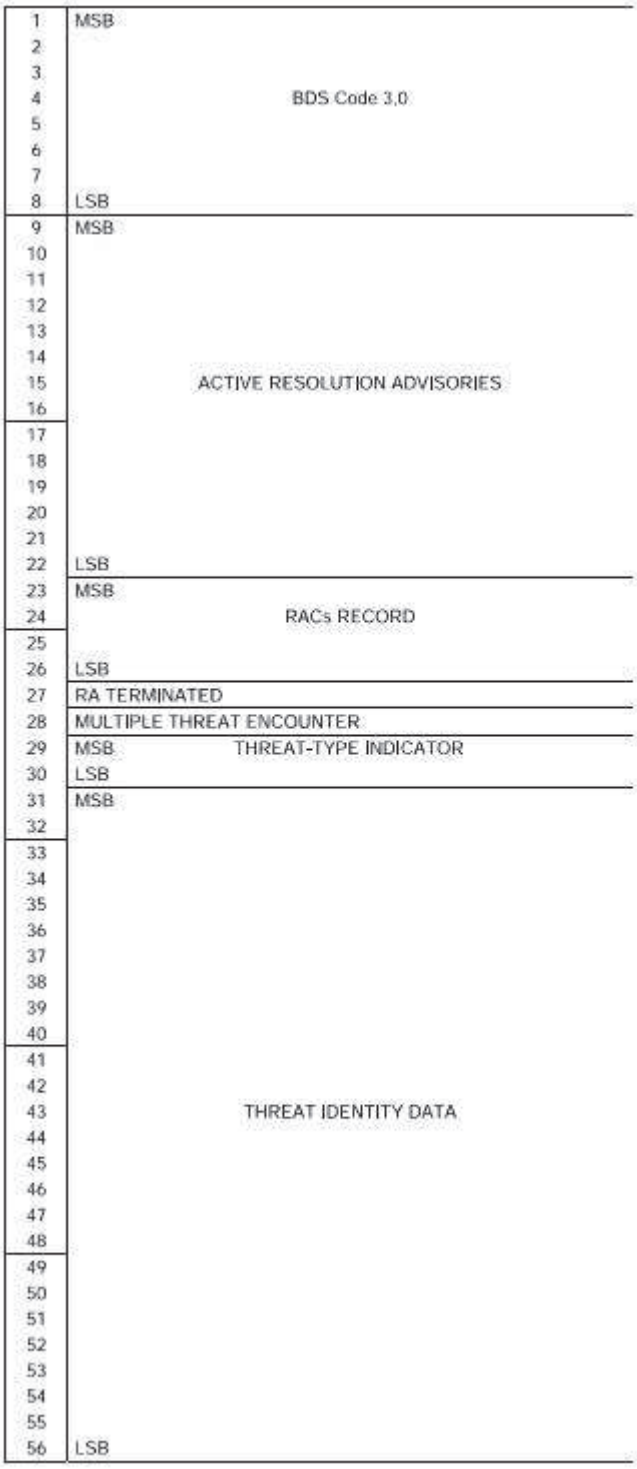


Figure 46. BDS code $3,0_{16}$ - ACAS active resolution advisory

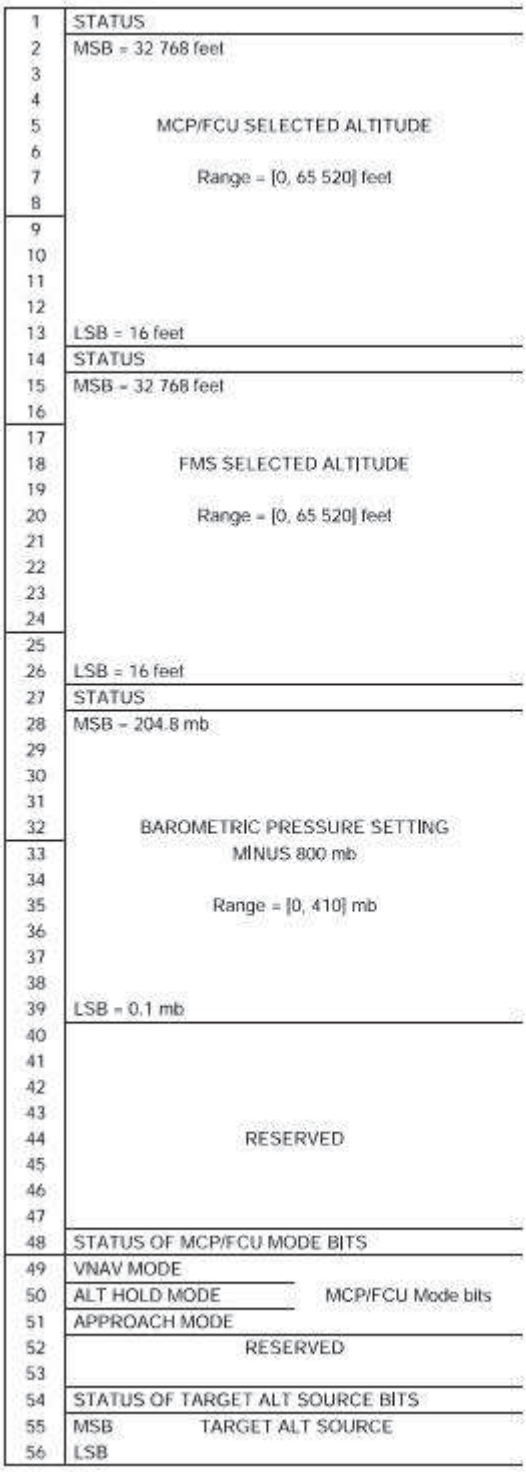


Figure 47. BDS code $4,0_{16}$ - Selected Vertical intention

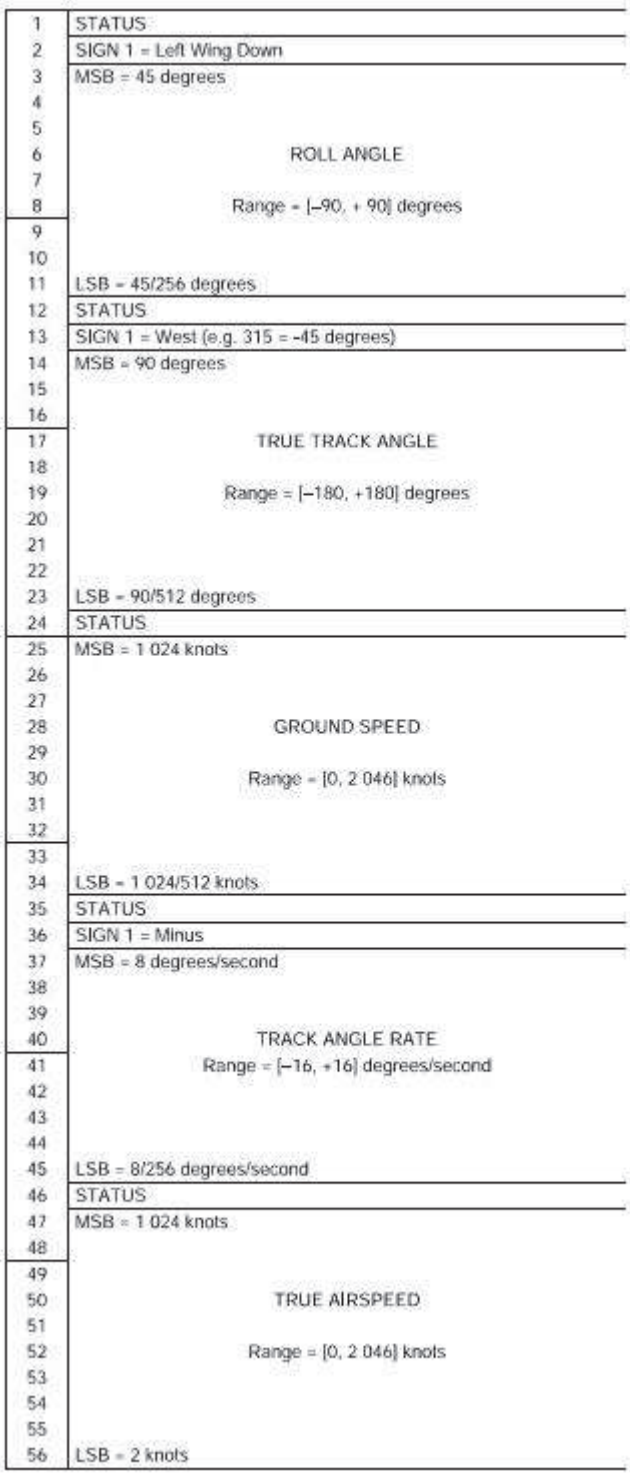


Figure 48. BDS code $5,0_{16}$ - Track and turn report

1	STATUS
2	SIGN 1=West (e.g. 315 = –45 degrees)
3	MSB = 90 degrees
4	
5	
6	MAGNETIC HEADING
7	
8	Range = [–180, +180] degrees
9	
10	
11	
12	LSB = 90/512 degrees
13	STATUS
14	MSB = 512 knots
15	
16	
17	INDICATED AIRSPEED
18	
19	Range = [0, 1023] knots
20	
21	
22	
23	LSB = 1 knot
24	STATUS
25	MSB = 2.048 MACH
26	
27	
28	MACH
29	
30	Range = [0, 4.092] MACH
31	
32	
33	
34	LSB = 2.048/512 MACH
35	STATUS
36	SIGN 1 = Below
37	MSB = 8 192 feet/minute
38	
39	
40	BAROMETRIC ALTITUDE RATE
41	
42	Range = [–16 384, +16 352] feet/minute
43	
44	
45	LSB = 8 192/256 = 32 feet/minute
46	STATUS
47	SIGN 1 = Below
48	MSB = 8 192 feet/minute
49	
50	
51	INERTIAL VERTICAL VELOCITY
52	
53	Range = [–16 384, +16 352] feet/minute
54	
55	
56	LSB = 8 192/256 = 32 feet/minute

Figure 49. BDS code $6,0_{16}$ - Heading and speed report

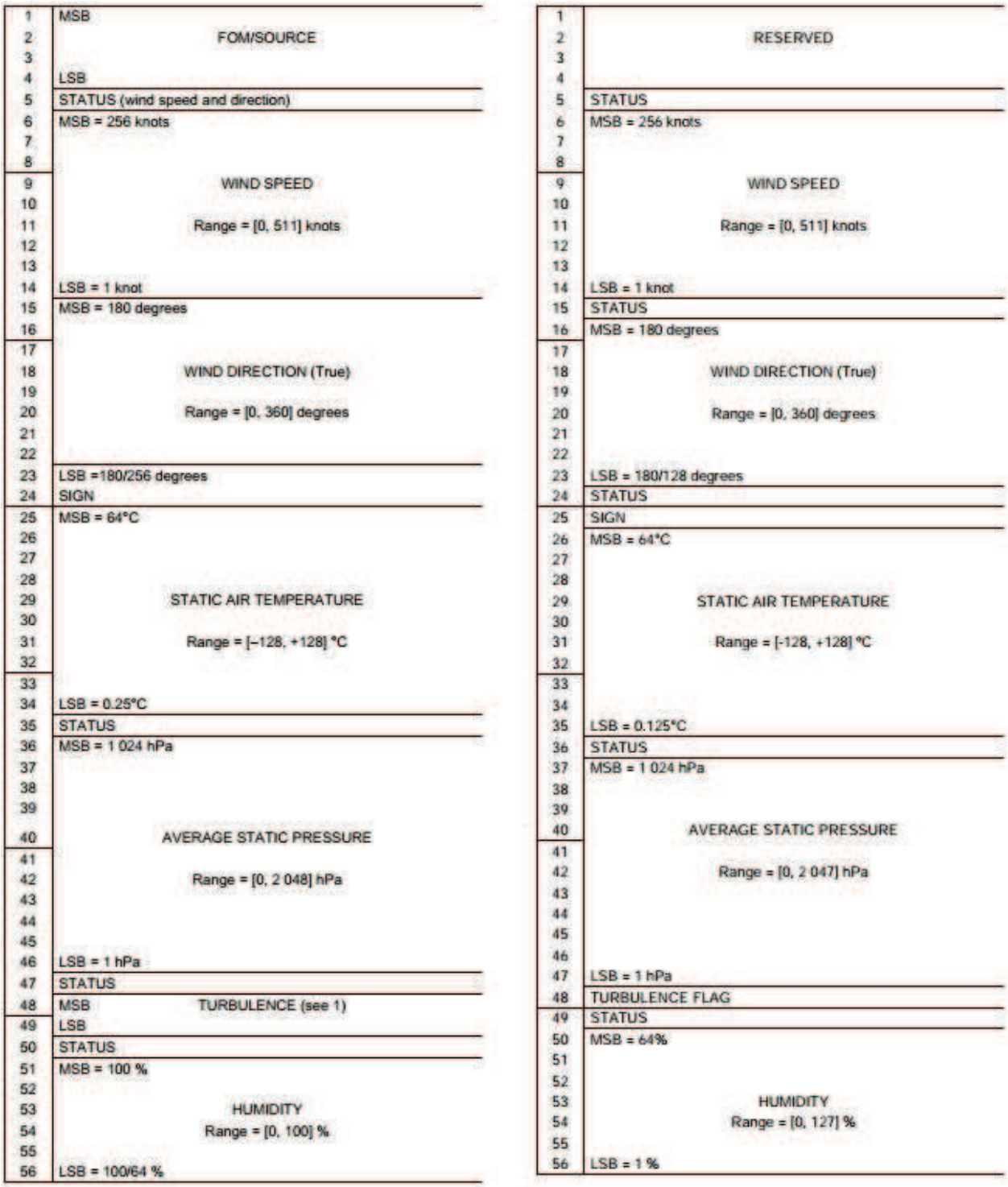


Figure 50. BDS code $4,4_{16}$ - Meteorological routine air report current version (right), [5] future version (left) [12]

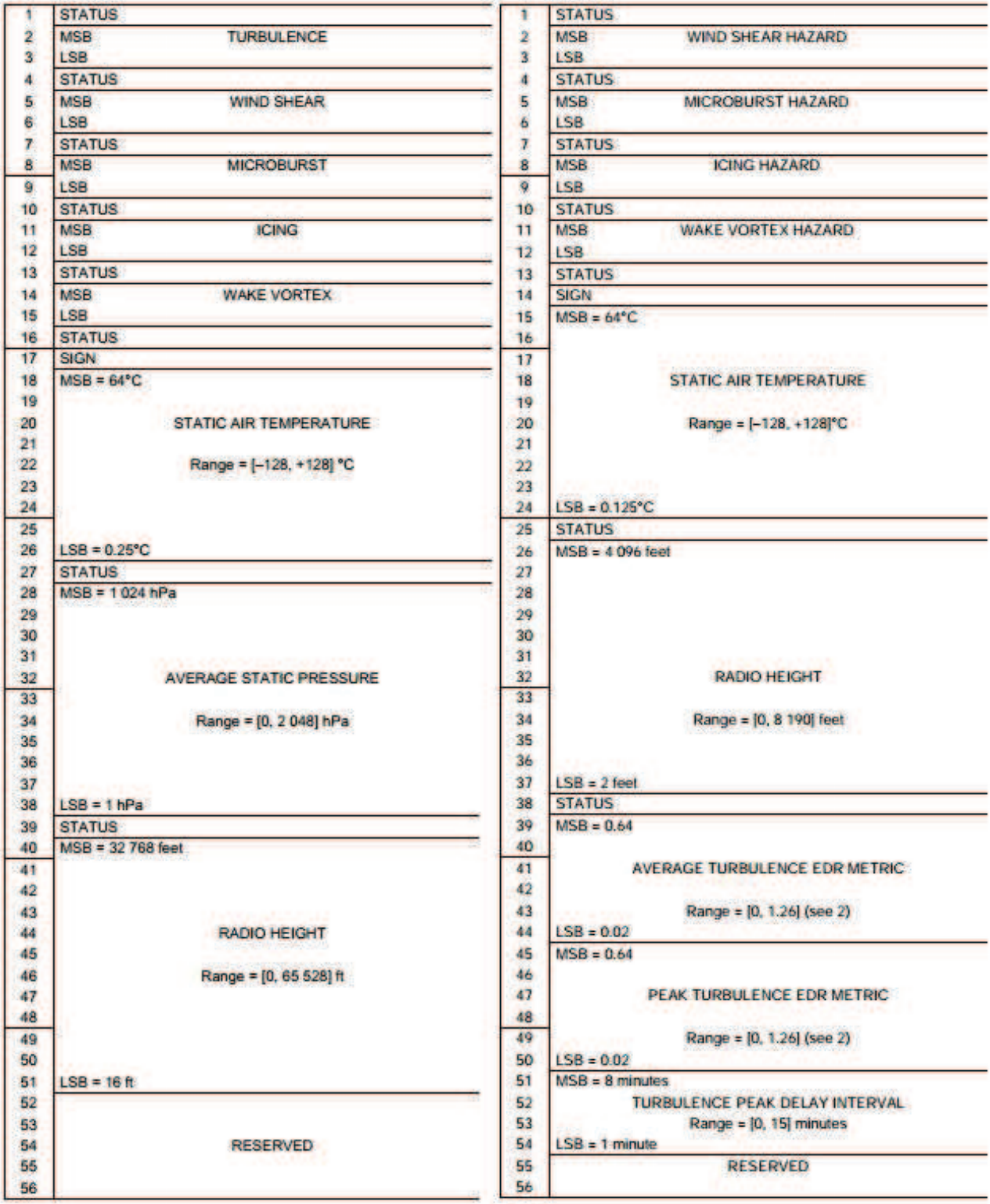


Figure 51. BDS code $4,5_{16}$ - Meteorological hazard report current version (right), [5] future version (left) [12]

Appendix B. Initial Data Processing MATLAB Script for Data from ADS-B Receiver

```
clear all
close all

addpath('./pomocne_fce');

%Data collection
%%
        nazev='1707180200'; %file name, which shall be analysed
        filename = ['./Archiv/',nazev,'.dat']; %file access
        delimiter = ';';

        %% Format string for each line of text:
        %   column1: double (%f)
        %   column2: double (%f)
        %   column3: text (%s)
        %   column4: text (%s)
        formatSpec = '%f%f%s%s%[^\n\r]';

        %% Open the text file.
        fileID = fopen(filename,'r');
        dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, 'EmptyValue' ,NaN,
'ReturnOnError', false);
        fclose(fileID);

        %% Allocate imported array to column variable names
        Unix_time = dataArray{:, 1};
        prijimac = dataArray{:, 2};
        time = dataArray{:, 3};
        Message = dataArray{:, 4};

        %% Clear temporary variables
        clearvars filename delimiter formatSpec fileID dataArray ans;

        mm=size(Unix_time,1);

%Defining variables of initial message processing
        DF=NaN(mm,1); %Downlink format
        CRC=cell(mm,1); %Cyclic redundancy check
%         Adress24=cell(mm,1);% ICAO address hexadecimal
        MessageBIN=cell(mm,1);% Message in binary code
        BDS=cell(mm,1); %BDS message in binary code
        BDSmessages=[];
%         Time_s=NaN(mm,1); %time saving in s
%         Time_ns=NaN(mm,1); %time saving in ns

%Data filtration
        for k=1:mm
           delka=length(Message{k,1}); % message lenght
           if delka==28
                   zpravaBIN=hex2bin2(Message{k,1}); % tested message
%                    timeBIN=hex2bin2(time{k,1});
%                    time_s=polyval(timeBIN(1,1:18),2);%binary to decimal transformation,
second till midnight
%                    time_ns=polyval(timeBIN(1,19:48),2);%nanoseconds

                   if zpravaBIN(1,1:5)==[1 0 1 0 0] %if DF20
                       DF(k,1)=20;
                       BDS{k,1}=zpravaBIN(1,33:88);
                       [Adress,Adress_hex]=crc_division4(zpravaBIN);
                       CRC{k,1}=Adress_hex;
%                        Adress24{k,1}=Adress_hex;
                       MessageBIN{k,1}=zpravaBIN;
%                        Time_s(k,1)=time_s;
%                        Time_ns(k,1)=time_ns;

                   elseif zpravaBIN(1,1:5)==[1 0 1 0 1] %if DF21
                       DF(k,1)=21;
                       BDS{k,1}=zpravaBIN(1,33:88);
                       [Adress,Adress_hex]=crc_division4(zpravaBIN);
                       CRC{k,1}=Adress_hex;
%                        Adress24{k,1}=Adress_hex;
```

```
                        MessageBIN{k,1}=zpravaBIN;
%                         Time_s(k,1)=time_s;
%                         Time_ns(k,1)=time_ns;
                    end

                    CRC_check=(length(CRC{k,1})); %for length check
                    if CRC_check==6 %CRC length check
                       Zero_CRC=strcmp(CRC{k,1},'000000'); %erroneous CRC exclusion
                       if Zero_CRC==0;
                       BDSmessages=[BDSmessages;BDS{k,1}]; %filtrated BDS message saving
                       end
                    end
            end
        end

file=['.\Test\',nazev,];
save(file,'BDSmessages'); %variables saving
```

Appendix C. Initial Data Processing MATLAB Script for ATN CR Data

```
close all
clear all

addpath('./pomocne_fce');
%% Import data from text file.
% Script for importing data from the following text file:
%
%    E:\Project\Data Frei\dump3030.20170505.1625.05.xml
%
% To extend the code to different selected data or a different text file,
% generate a function instead of a script.

% Auto-generated by MATLAB on 2017/09/13 10:17:05

%% Initialize variables.
filename = 'E:\Project\Data ATN\dump3030.20170505.1625.05.xml';
delimiter = {'<','>','/'};

%% Read columns of data as text:
% For more information, see the TEXTSCAN documentation.
formatSpec = '%q%q%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, 'Delimiter', delimiter, 'MultipleDelimsAsOne', true,
'TextType', 'string',  'ReturnOnError', false);

%% Close the text file.
fclose(fileID);

%% Convert the contents of columns containing numeric text to numbers.
% Replace non-numeric text with NaN.
raw = repmat({''},length(dataArray{1}),length(dataArray)-1);
for col=1:length(dataArray)-1
    raw(1:length(dataArray{col}),col) = mat2cell(dataArray{col}, ones(length(dataArray{col}),
1));
end
numericData = NaN(size(dataArray{1},1),size(dataArray,2));


%% Split data into numeric and string columns.
rawNumericColumns = {};
rawStringColumns = string(raw(:, [1,2]));


%% Replace non-numeric cells with NaN
R = cellfun(@(x) ~isnumeric(x) && ~islogical(x),rawNumericColumns); % Find non-numeric cells
rawNumericColumns(R) = {NaN}; % Replace non-numeric cells

%% Create output variable
% Asterix_data = table;
% Data = rawStringColumns(:, 1);
Data = rawStringColumns(:, 1);
Value = rawStringColumns(:, 2);

%% Clear temporary variables
clearvars filename delimiter formatSpec fileID dataArray ans raw col numericData
rawNumericColumns rawStringColumns R;

%Data filtration
s=size(Data,1); %number of rows
BDS1 = NaN(s,1);
BDS2 = NaN(s,1);
RawMessage = cell(s,1);
MessageBIN = cell(s,1);
BDSmessages=[];
```

```
BDSasterix=[];

for k=1:s
    if Data(k,1)== 'MBdata'
       RawMessage{k,1}= Value(k,1);
       Message=RawMessage{k,1};
       l=k+1;
       m=k+2;
       MessageBIN{k,1}=hex2bin2(Message{1,1});
       BDSmessages=[BDSmessages;MessageBIN{k,1}];
        if Data(l,1)=='DSB1'
        BDS1(k,1)= Value(l,1);
            if Data(m,1)=='DSB2'
            BDS2(k,1)= Value(m,1);
            end
        end
        BDSasterix=[BDSasterix;BDS1(k,1) BDS2(k,1)];
    end
end

nazev='AsterixData';
file=['.\Test\',nazev,];
save(file,'BDSmessages','BDSasterix'); %variables saving
```

Appendix D. Heuristic Algorithm MATLAB Script

```
close all
clear all

% %loading ADS-B receivers data
% nazev='1708010200'; %file name, which shall be analysed
% filename = ['./Test/',nazev,'.mat']; %file access
% load (filename,'BDSmessages');

%loading ATN data
nazev='ThirdAsterixData'; %file name, which shall be analysed
filename = ['./Test/',nazev,'.mat']; %file access
load (filename,'BDSmessages','BDSasterix');

%defining variables
global Tested_message; %global variable for used function
Tested_message=[]; %defining global variable
s=size(BDSmessages,1); %number of BDS rows
BDS_number_guess=zeros(s,9); %saving of BDS number, distinquishing of 9 different registers
BDS_guess_counter=zeros(s,1); %counting assign register number
BDS_final=cell(s,1); %final BDS register number

%BDS assigning criteria testing
for i=1:s;
  Tested_message=BDSmessages(i,:); %tested message

  %Exluding zeros messages
  ZeroCounter=sum(BDSmessages(i,1:56)==0); %number of zeros in a message
  if ZeroCounter(1,1)<=55; %no testing of zeros messages

  %Testing registers with BDS1 and BDS2 code
  BDSidentifier=bi2de(BDSmessages(i,1:8),'left-msb'); %saving BDS1 and BDS2 code in decimal
scale

    %BDS 1,0
    if BDSidentifier==16; %hexadecimal 1,0 in decimal scale
        BDSOneZeroGuess=BDSmessages(i,:); %saving for following testing
        OneZeroField(1,1)=bi2de(BDSOneZeroGuess(17:20),'left-msb'); %reserved values in field
        OneZeroField(1,2)=bi2de(BDSOneZeroGuess(21:23),'left-msb'); %required values in field
        OneZeroField(1,3)=bi2de(BDSOneZeroGuess(29:32),'left-msb'); %reserved values in field
            if OneZeroField(1,1)==0 && (OneZeroField(1,2)==0 || OneZeroField(1,2)==3 ||
OneZeroField(1,2)==4 || OneZeroField(1,2)==5) && OneZeroField(1,3)<=6 &&
BDSOneZeroGuess(1,33)==1 && BDSOneZeroGuess(1,35)==1 && ZeroCounter(1,1)>43; %conditions for
BDS 1,0
               BDS_number_guess(i,1)=10; %BDS number saving
            end

    %BDS 2,0
    elseif BDSidentifier==32; %hexadecimal 2,0 in decimal scale
            BDSTwoZeroGuess=BDSmessages(i,:); %saving for following testing
            IDField(1,1)=bi2de(BDSTwoZeroGuess(1,9:14),'left-msb'); %field of characters
            IDField(1,2)=bi2de(BDSTwoZeroGuess(1,15:20),'left-msb'); %field of characters
            IDField(1,3)=bi2de(BDSTwoZeroGuess(1,21:26),'left-msb'); %field of characters
            IDField(1,4)=bi2de(BDSTwoZeroGuess(1,27:32),'left-msb'); %field of characters
            IDField(1,5)=bi2de(BDSTwoZeroGuess(1,33:38),'left-msb'); %field of characters
            IDField(1,6)=bi2de(BDSTwoZeroGuess(1,39:44),'left-msb'); %field of characters
            IDField(1,7)=bi2de(BDSTwoZeroGuess(1,45:50),'left-msb'); %field of characters
            IDField(1,8)=bi2de(BDSTwoZeroGuess(1,51:56),'left-msb'); %field of characters
            NoDefined=[0 27:31 33:47]; %not allowed values
            for l=1:length(NoDefined); %loop of not allowed values
                if IDField (1,:)~= NoDefined(l); %testing of allowed values
                    if IDField (1,1)~=32 || IDField (1,2)~=32 || IDField (1,3)~=32 || IDField
(1,4)~=32; %testing intervening spaces
                    BDS_number_guess(i,2)=20; %BDS number saving
                    end
                end
            end

    %BDS 3,0
    elseif BDSidentifier==48; %hexadecimal 3,0 in decimal scale
        ThreeZeroField(1,1)=bi2de(BDSmessages(i,16:22),'left-msb');
        ThreeZeroField(1,2)=bi2de(BDSmessages(i,10:22),'left-msb');
        ThreeZeroField(1,3)=bi2de(BDSmessages(i,55:56),'left-msb');
        ThreeZeroField(1,4)=bi2de(BDSmessages(i,29:30),'left-msb');
```

```
        if (BDSmessages(i,9)==1 && BDSmessages(i,28)==0) || (BDSmessages(i,9)==1 &&
BDSmessages(i,28)==1) || (BDSmessages(i,9)==0 && BDSmessages(i,28)==1) &&
ThreeZeroField(1,1)==0;
            if ThreeZeroField(1,4)==1 && ThreeZeroField(1,3)==0;
                BDS_number_guess(i,3)=30;
            end
        elseif BDSmessages(i,9)==0 && BDSmessages(i,28)==0 && ThreeZeroField(1,2)==0;
            if ThreeZeroField(1,4)==1 && ThreeZeroField(1,3)==0;
                BDS_number_guess(i,3)=30;
            end
        end
    end

     %BDS 1,7
     OneSevenField(1,1)=bi2de(BDSmessages(i,26:56),'left-msb'); %reserved bits
     if OneSevenField(1,1)==0 && ZeroCounter(1,1)>=40; %conditions for BDS 1,7
     BDS_number_guess(i,9)=17; %BDS number saving
     end

    %Defining variables for EHS and MRAR registers
    %significant bits for BDS 4,0
    FourZeroStatusBit=[BDSmessages(i,1) BDSmessages(i,14) BDSmessages(i,27) BDSmessages(i,48)
BDSmessages(i,54)];
    FourZeroStartBit=[2 15 28 49 55];
    FourZeroEndBit=[13 26 39 51 56];

    %significant bits for BDS 5,0
    FiveZeroStatusBit=[BDSmessages(i,1) BDSmessages(i,12) BDSmessages(i,24) BDSmessages(i,35)
BDSmessages(i,46)];
    FiveZeroStartBit=[2 13 25 36 47];
    FiveZeroEndBit=[11 23 34 45 56];

    %significant bits for BDS 6,0
    SixZeroStatusBit=[BDSmessages(i,1) BDSmessages(i,13) BDSmessages(i,24) BDSmessages(i,35)
BDSmessages(i,46)];
    SixZeroStartBit=[2 14 25 36 47];
    SixZeroEndBit=[12 23 34 45 56];

    %significant bits for BDS 4,4
    FourFourStatusBit=[BDSmessages(i,5) BDSmessages(i,5) BDSmessages(i,24) BDSmessages(i,35)
BDSmessages(i,47) BDSmessages(i,50)];
    FourFourStartBit=[6 15 25 36 48 51];
    FourFourEndBit=[14 23 34 46 49 56];

    %significant bits for BDS 4,5
    FourFiveStatusBit=[BDSmessages(i,1) BDSmessages(i,4) BDSmessages(i,7) BDSmessages(i,10)
BDSmessages(i,13) BDSmessages(i,16) BDSmessages(i,27) BDSmessages(i,39)];
    FourFiveStartBit=[2 5 8 11 14 17 28 40];
    FourFiveEndBit=[3 6 9 12 15 26 38 51];

    %Testing EHS registers
    %BDS 4,0
    FourZeroField(1,1)=bi2de(BDSmessages(i,29:31),'left-msb'); %required values in field
    FourZeroField(1,2)=bi2de(BDSmessages(i,40:47),'left-msb'); %reserved bits
    FourZeroField(1,3)=bi2de(BDSmessages(i,51:53),'left-msb'); %reserved bits
    FourZero_OneCounter(1,1)=sum(BDSmessages(i,1:56)==1); %number of ones
        if FourZeroField(1,1)==0 && FourZeroField(1,2)==0 && FourZeroField(1,3)==0 &&
FourZero_OneCounter(1,1)<27; %testing conditions for BDS 4,0
          DecodedFourZeroGuess=Decoding(FourZeroStatusBit,FourZeroStartBit,FourZeroEndBit);
%status bit testing and decoding function
            if DecodedFourZeroGuess(1,1)<=2950; %maximum altitude
                if DecodedFourZeroGuess(1,2)<=2950; %maximum altitude
                    if DecodedFourZeroGuess(1,3)==0 || (DecodedFourZeroGuess(1,3)<=2220 &&
DecodedFourZeroGuess(1,3)>=2080); %maximum barometric pressure
                        if DecodedFourZeroGuess(1,4)==0 || DecodedFourZeroGuess(1,4)==2 ||
DecodedFourZeroGuess(1,4)==4; %possible acquired values
                            if DecodedFourZeroGuess(1,5)<=3; %possible acquired values
                        BDS_number_guess(i,4)=40; %BDS number saving
                            end
                        end
                    end
                end
            end
        end

    %BDS 5,0
```

```
    if BDSmessages(i,25)==0 && BDSmessages(i,47)==0 && BDSmessages(i,48)==0; %testing of the
most significant bits
       DecodedFiveZeroGuess=Decoding(FiveZeroStatusBit,FiveZeroStartBit,FiveZeroEndBit);
%status bit testing and decoding function
        if DecodedFiveZeroGuess(1,1)<=170 || (DecodedFiveZeroGuess(1,1)>=850 &&
DecodedFiveZeroGuess(1,1)<=1024); %maximum roll angle, distinguishing condition
            if DecodedFiveZeroGuess(1,2)<=2048; %true track angle
                if DecodedFiveZeroGuess(1,3)<=270; %maximum ground speed
                    if DecodedFiveZeroGuess(1,4)<=1024; %maximum track angle
                        if DecodedFiveZeroGuess(1,5)<=270; %maximum true airspeed
                           BDS_number_guess(i,5)=50; %BDS number saving
                        end
                    end
                end
            end
        end
    end

     %BDS 6,0
     if BDSmessages(i,14)==0 && BDSmessages(i,25)==0 && BDSmessages(i,26)==0 &&
BDSmessages(i,35)==1; %testing of the most significant bits
        DecodedSixZeroGuess=Decoding(SixZeroStatusBit,SixZeroStartBit,SixZeroEndBit); %status
bit testing and decoding function
        if DecodedSixZeroGuess(1,1)<=2048; %magnetic heading
            if DecodedSixZeroGuess(1,2)<=540; %maximum indicated airspeed
                if DecodedSixZeroGuess(1,3)<=230; %maximum speed
                    if DecodedSixZeroGuess(1,4)<=130 || (DecodedSixZeroGuess(1,4)>=884 &&
DecodedSixZeroGuess(1,4)<=1024); %maximum barometric altitude rate
                        if DecodedSixZeroGuess(1,5)<=124 || (DecodedSixZeroGuess(1,5)>=900 &&
DecodedSixZeroGuess(1,5)<=1024); %maximum inertial vertical velocity
                        BDS_number_guess(i,6)=60; %BDS number saving
                        end
                    end
                end
            end
         end
     end

     %Testing MRAR registers
     %BDS 4,4
     FourZeroField(1,1)=bi2de(BDSmessages(i,1:4),'left-msb'); %FOM/SOURCE field decoding
        if (FourZeroField(1,1)==1 || FourZeroField(1,1)==2 || FourZeroField(1,1)==3 ||
FourZeroField(1,1)==4); %FOM/SOURCE field testing
           DecodedFourFourGuess=Decoding(FourFourStatusBit,FourFourStartBit,FourFourEndBit);
%status bit testing and decoding function
            if DecodedFourFourGuess(1,1)<=80; %maximum wind speed
                if DecodedFourFourGuess(1,2)<=512; %wind direction
                    if DecodedFourFourGuess(1,3)<=240 || (DecodedFourFourGuess(1,3)>=704 &&
DecodedFourFourGuess(1,3)<=1024); %static air temperature
                        if DecodedFourFourGuess(1,4)<=1080; %average static pressure
                            if DecodedFourFourGuess(1,5)<=3; %turbulence status
                                if DecodedFourFourGuess(1,6)<=64; %maximum humidity
                                    BDS_number_guess(i,7)=44; %BDS number saving
                                end
                            end
                        end
                    end
                end
            end
        end

     %BDS 4,5
         if BDS_number_guess(i,:)==0; %testing only if no other BDS number has been assigned
            FiveZeroField(1,1)=bi2de(BDSmessages(i,52:56),'left-msb'); %reserved field
decoding
            OnesValues=sum(BDSmessages(i,1:15)==1); %hazardous values set at 1 counting
            ZerosValues=sum(BDSmessages(i,1:15)==0); %hazardous values set at 0 counting
                if FiveZeroField(1,1)==0 && OnesValues<=10 && OnesValues>=4; %reserved field
and hazardeous content testing

DecodedFourFiveGuess=Decoding(FourFiveStatusBit,FourFiveStartBit,FourFiveEndBit); %status bit
testing and decoding function
                    if DecodedFourFiveGuess(1,1)<=3; %turbulence
                        if DecodedFourFiveGuess(1,2)<=3; %wind shear
                            if DecodedFourFiveGuess(1,3)<=3; %microburst
                                if DecodedFourFiveGuess(1,4)<=3; %icing
                                    if DecodedFourFiveGuess(1,5)<=3; %wake vortex
```

```
                                        if DecodedFourFiveGuess(1,6)<=240 ||
(DecodedFourFiveGuess(1,6)>=704 && DecodedFourFiveGuess(1,6)<=1024); %static air temperature
                                            if DecodedFourFiveGuess(1,7)<=1080 &&
DecodedFourFiveGuess(1,7)>0; %maximum static pressure
                                                if DecodedFourFiveGuess(1,8)<=2950 &&
DecodedFourFiveGuess(1,8)>0; %maximum altitude
                                                    BDS_number_guess(i,8)=45; %BDS number
saving
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
    end
        BDS_guess_counter(i,1)=sum(BDS_number_guess(i,1:9)==0); %BDS number counting
end

%final BDS number assignment
for i=1:s;
    %Unknown BDS
    if BDS_guess_counter(i,1)==9; %another BDS register
        BDS_final{i,1}='Unknown'; %final BDS number assignment

    %BDS with sufficient testing (BDS 1,0 and BDS 2,0)
    elseif BDS_number_guess(i,1)==10;
        BDS_final{i,1}=10; %final BDS number assignment

    elseif BDS_number_guess(i,2)==20;
        BDS_final{i,1}=20; %final BDS number assignment

    elseif BDS_guess_counter(i,1)==8; %only one assigned BDS number
        for j=1:9;
            if BDS_number_guess(i,j)~=0; %choosing the one assigned BDS number
                BDS_final{i,1}=BDS_number_guess(i,j); %final BDS number assignment
            end
        end
        if BDS_number_guess(i,8)==45; %BDS 45 is not trasmitted
            BDS_final{i,1}='Unknown';
        end

    elseif BDS_guess_counter(i,1)==7; %two assigned BDS registers

        %40 and 50 conflict
        if BDS_number_guess(i,4)==40 && BDS_number_guess(i,5)==50;
            BDS_final{i,1}=40; %final BDS number assignment
        end

        %44 and 17 conflict
        if BDS_number_guess(i,7)==44 && BDS_number_guess(i,9)==17;
            BDS_final{i,1}=17; %final BDS number assignment
        end

        %50 and 17 conflict
        if BDS_number_guess(i,5)==50 && BDS_number_guess(i,9)==17;
            BDS_final{i,1}=17; %final BDS number assignment
        end

        %50 and 60 conflict
        if BDS_number_guess(i,5)==50 && BDS_number_guess(i,6)==60;
            BDS_final{i,1}=60; %final BDS number assignment
        end

        %40 and 60 conflict
        if BDS_number_guess(i,4)==40 && BDS_number_guess(i,6)==60;
            OnesCounter=sum(BDSmessages(i,1:56)==1); %number of ones
            if OnesCounter<=15; %condition for assignment
                BDS_final{i,1}=40; %final BDS number assignment
            else BDS_final{i,1}=60; %final BDS number assignment
            end
        end

        %40 and 17 conflict
```

```
        if BDS_number_guess(i,4)==40 && BDS_number_guess(i,9)==17;
            OnesCounter=sum(BDSmessages(i,1:56)==1); %number of ones
            if OnesCounter<=2 && BDSmessages(i,7)==1 && BDSmessages(i,16)==0; %condition for
assignment
                BDS_final{i,1}=17; %final BDS number assignment
            else BDS_final{i,1}=40; %final BDS number assignment
            end
        end

    elseif BDS_guess_counter(i,1)==6; %three assigned BDS registers

        %40, 50 and 17 conflict
        if BDS_number_guess(i,4)==40 && BDS_number_guess(i,5)==50 &&
BDS_number_guess(i,9)==17;
            OnesCounter=sum(BDSmessages(i,1:56)==1); %number of ones
            if OnesCounter>=4;
            BDS_final{i,1}=17; %final BDS number assignment
            else BDS_final{i,1}='Unknown';
            end
        end

        %40, 44 and 17 conflict
        if BDS_number_guess(i,4)==40 && BDS_number_guess(i,7)==44 &&
BDS_number_guess(i,9)==17;
            BDS_final{i,1}='Unknown'; %final BDS number assignment
        end
    end
end

%BDS saving
%variables for BDS messages counting
a=0;
b=0;
c=0;
d=0;
e=0;
f=0;
g=0;
h=0;
j=0;

%counting number of particular BDS registers
for i=1:s;
    if BDS_final{i,1}==10;
       a=a+1;
    elseif BDS_final{i,1}==20;
       b=b+1;
    elseif BDS_final{i,1}==30;
       c=c+1;
    elseif BDS_final{i,1}==40;
       d=d+1;
    elseif BDS_final{i,1}==50;
       e=e+1;
    elseif BDS_final{i,1}==60;
       f=f+1;
    elseif BDS_final{i,1}==44;
       g=g+1;
    elseif BDS_final{i,1}==45;
       h=h+1;
    elseif BDS_final{i,1}==17;
       j=j+1;
    end
end

%defining variables for saving
BDSOneZero=zeros(a,56);
BDSTwoZero=zeros(b,56);
BDSThreeZero=zeros(c,56);
BDSFourZero=zeros(d,56);
BDSFiveZero=zeros(e,56);
BDSSixZero=zeros(f,56);
BDSFourFour=zeros(g,56);
BDSFourFive=zeros(h,56);
BDSOneSeven=zeros(j,56);

a=1;
b=1;
```

```
c=1;
d=1;
e=1;
f=1;
g=1;
h=1;
j=1;

%saving corresponding messages
for i=1:s;
    if BDS_final{i,1}==10;
        BDSOneZero(a,:)=BDSmessages(i,:);
        a=a+1;
    elseif BDS_final{i,1}==20;
        BDSTwoZero(b,:)=BDSmessages(i,:);
        b=b+1;
    elseif BDS_final{i,1}==30;
        BDSThreeZero(c,:)=BDSmessages(i,:);
        c=c+1;
    elseif BDS_final{i,1}==40;
        BDSFourZero(d,:)=BDSmessages(i,:);
        d=d+1;
    elseif BDS_final{i,1}==50;
        BDSFiveZero(e,:)=BDSmessages(i,:);
        e=e+1;
    elseif BDS_final{i,1}==60;
        BDSSixZero(f,:)=BDSmessages(i,:);
        f=f+1;
    elseif BDS_final{i,1}==44;
        BDSFourFour(g,:)=BDSmessages(i,:);
        g=g+1;
    elseif BDS_final{i,1}==45;
        BDSFourFive(h,:)=BDSmessages(i,:);
        h=h+1;
    elseif BDS_final{i,1}==45;
        BDSOneSeven(j,:)=BDSmessages(i,:);
        j=j+1;
    end
end

%saving all registers to MAT-file
name=('ThirdAsterixIdentifiedBDS');
file=['.\Meteo\',name,'.mat'];
save(file,'BDSOneZero','BDSTwoZero','BDSThreeZero','BDSFourZero','BDSFiveZero','BDSSixZero','B
DSFourFour','BDSFourFive','BDSOneSeven');

%saving MRAR registers to MAT-file
name=('ThirdAsterixMeteoBDS');
file=['.\Meteo\',name,'.mat'];
save(file,'BDSFourFour','BDSFourFive');

% name=('ADSBIdentifiedBDS');
% file=['.\Meteo\',name,'.mat'];
%
save(file,'BDSOneZero','BDSTwoZero','BDSThreeZero','BDSFourZero','BDSFiveZero','BDSSixZero','B
DSFourFour','BDSFourFive','BDSOneSeven');
%
% name=('ADSBMeteoBDS');
% file=['.\Meteo\',name,'.mat'];
% save(file,'BDSFourFour','BDSFourFive');
```

Appendix E. Meteorological Decoder MATLAB Script

```
close all
clear all

nazev='AsterixMeteoBDS'; %file name, which shall be analysed
filename = ['.\Meteo\',nazev,'.mat']; %file access
load (filename,'BDSFourFour','BDSFourFive');

% BDS 4,4 decoding
m=size(BDSFourFour,1); %number of rows

%significant bits for BDS 4,4
FourFourStartBit=[1 6 15 25 36 48 51];
FourFourEndBit=[4 14 23 34 46 49 56];
DecodedBDSFourFour=cell(m,1);

for i=1:m;
    %Saving of tested message
    TestedMessage=BDSFourFour(i,:);

    %Decoded message in decimal scale
    DecodedFourFour=DecodingMeteo(TestedMessage,FourFourStartBit,FourFourEndBit); %status bit
testing and decoding function

    %Recalculation of wind direction
    DecodedFourFour(1,3)=DecodedFourFour(1,3).*(180/256);

    %Recalculation of temperature
    if TestedMessage(1,25)==0; %positive values
          DecodedFourFour(1,4)=(DecodedFourFour(1,4).*0.25);
    elseif TestedMessage(1,25)==1; %negative values
          DecodedFourFour(1,4)=((DecodedFourFour(1,4)-1024).*0.25);
    end

    %Recalculation of humidity
    DecodedFourFour(1,7)=DecodedFourFour(1,7).*(64/100);

    %Convertion to cell
    DecodedFourFour=num2cell(DecodedFourFour);

    %FOM/SOURCE decoding
    FOM_SOURCEoptions={'Invalid','INS','GNSS','DME/DME','VOR/DME','Reserved'};

    if DecodedFourFour{1,1}==0;
        DecodedFourFour{1,1}=FOM_SOURCEoptions(1,1);
    elseif DecodedFourFour{1,1}==1;
        DecodedFourFour{1,1}=FOM_SOURCEoptions(1,2);
    elseif DecodedFourFour{1,1}==2;
        DecodedFourFour{1,1}=FOM_SOURCEoptions(1,3);
    elseif DecodedFourFour{1,1}==3;
        DecodedFourFour{1,1}=FOM_SOURCEoptions(1,4);
    elseif DecodedFourFour{1,1}==4;
        DecodedFourFour{1,1}=FOM_SOURCEoptions(1,5);
    else DecodedFourFour{1,1}=FOM_SOURCEoptions(1,6);
    end

    %TURBULENCE decoding
    if TestedMessage(1,47)==0;
    DecodedFourFour{1,5}=0;
    else
    DecodedFourFour{1,5}=ValueInterpretation(DecodedFourFour{1,5});
    end

    %all messages saving
    DecodedBDSFourFour{i}=DecodedFourFour;
end

% BDS 4,5 decoding
n=size(BDSFourFive,1); %number of rows

%significant bits for BDS 4,4
FourFiveStartBit=[2 5 8 11 14 17 28 40];
FourFiveEndBit=[3 6 9 12 15 26 38 51];
DecodedBDSFourFive=cell(n,1);
```

```
for i=1:n; %No status value
    %Saving of tested message
    TestedMessage=BDSFourFive(i,:);

    %Decoded message in decimal scale
    DecodedFourFive=DecodingMeteo(TestedMessage,FourFiveStartBit,FourFiveEndBit); %status bit
testing and decoding function

    %Recalculation of temperature
    if TestedMessage(1,17)==0; %positive values
          DecodedFourFive(1,6)=(DecodedFourFive(1,6).*0.25);
    elseif TestedMessage(1,17)==1; %negative values
          DecodedFourFive(1,6)=((DecodedFourFive(1,6)-1024).*0.25);
    end

    %Recalculation of radio height
    DecodedFourFive(1,8)=DecodedFourFive(1,8)*16;

    %Convertion to cell
    DecodedFourFive=num2cell(DecodedFourFive);

    %TURBULENCE decoding
    if TestedMessage(1,1)==0;
    DecodedFourFive{1,1}=0;
    else
    DecodedFourFive{1,1}=ValueInterpretation(DecodedFourFive{1,1});
    end

    %WIND_SHEAR decoding
    if TestedMessage(1,4)==0;
    DecodedFourFive{1,2}=0;
    else
    DecodedFourFive{1,2}=ValueInterpretation(DecodedFourFive{1,2});
    end

    %MICROBURST decoding
    if TestedMessage(1,7)==0;
    DecodedFourFive{1,3}=0;
    else
    DecodedFourFive{1,3}=ValueInterpretation(DecodedFourFive{1,3});
    end

    %ICING decoding
    if TestedMessage(1,10)==0;
    DecodedFourFive{1,4}=0;
    else
    DecodedFourFive{1,4}=ValueInterpretation(DecodedFourFive{1,4});
    end

    %WAKE VORTEX decoding
    if TestedMessage(1,13)==0;
    DecodedFourFive{1,5}=0;
    else
    DecodedFourFive{1,5}=ValueInterpretation(DecodedFourFive{1,5});
    end

    %all messages saving
    DecodedBDSFourFive{i}=DecodedFourFive;
end

name=('DecodedMeteo');
file=['.\Decoded\',name,'.mat'];
save(file,'DecodedBDSFourFour','DecodedBDSFourFour');
```

Appendix F. MATLAB Function for Hazardous Data Interpretation

```
function [ Interpretation ] = ValueInterpretation( Values);

m=size(Values,1);
InterpretationOptions={'Nil','Light','Moderate','Severe'};
Interpretation=cell(m,1);

for i=1:m;
    if Values(i,1)==0;
        Interpretation(i,1)=InterpretationOptions(1,1);
    elseif Values(i,1)==1;
        Interpretation(i,1)=InterpretationOptions(1,2);
    elseif Values(i,1)==2;
        Interpretation(i,1)=InterpretationOptions(1,3);
    elseif Values(i,1)==3;
        Interpretation{i,1}=InterpretationOptions(1,4);
    end
end
```