ARL-TN-0935 ● DEC 2018

US Army Research Laboratory

A MATLAB Inter-Range Instrumentation Group Chapter 10 Data Decoding Library

by Michael Don

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TN-0935 ● DEC 2018

US Army Research Laboratory

A MATLAB Inter-Range Instrumentation Group Chapter 10 Data Decoding Library

by Michael Don
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2018	Technical Note	July–August 2018

4. TITLE AND SUBTITLE
A MATLAB Inter-Range Instrumentation Group Chapter 10 Data Decoding Library

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Michael Don

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
US Army Research Laboratory
ATTN: RDRL-WML-F
Aberdeen Proving Ground, MD 21005

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-0935

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Military ranges record telemetry data in Inter-Range Instrumentation Group (IRIG) Chapter 10 format. The US Army Research Laboratory has written a custom library of MATLAB functions to serve its IRIG Chapter 10 decoding needs, namely the extraction of telemetry frames and timing from pulse code modulated and IRIG-B packets. In addition, the MATLAB implementation provides a convenient platform for further data analysis. First, a brief overview of the library is given. Then the functionality of the library is demonstrated with an example script that extracts telemetry data from a sample Chapter 10 file.

15. SUBJECT TERMS
IRIG Chapter 10, telemetry, data storage, data formatting, software library

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Michael Don
Unclassified	Unclassified	Unclassified	UU	25	19b. TELEPHONE NUMBER (Include area code) 410-306-0775

Contents

List of Figures

1. Introduction

Typically, the US Army Research Laboratory (ARL) uses its own telemetry receivers to record flight test data at military ranges. Both commercial and custom receivers have been employed for this purpose,[1–3] but military ranges make their own recording of the telemetry data in Inter-Range Instrumentation Group (IRIG) Chapter 10 format.[4] Open-source software exists to decode Chapter 10 files, but it does not provide all of the functionality necessary to fully extract ARL's telemetry data.[5] Commercial products exist as well, but it is difficult to justify the cost of this software based on the limited needs of ARL.[6] Thus ARL has written a custom library of MATLAB[7] functions to serve its IRIG Chapter 10 decoding needs, namely the extraction of telemetry frames and timing from pulse code modulated (PCM) and IRIG-B packets. In addition, the MATLAB implementation provides a convenient platform for further data analysis. First, a brief overview of the library is given. Then the functionality of the library is demonstrated with an example script that extracts telemetry data from a sample Chapter 10 file.

2. Library Overview

Chapter 10 files are organized into various types of packets.[4] PCM packets contain the telemetry data but also may contain randomized data, and the telemetry words are typically not aligned to the byte boundaries. IRIG-B timecode packets contain IRIG-B timing information. All packet headers contain a relative time counter. Therefore, in order to timestamp the PCM data, an IRIG-B packet is used to relate the relative counter to absolute IRIG time. The relative counters on the PCM packets can then be related to IRIG time, creating PCM timing information. This process is demonstrated in the example in Section 3.

The following functions provide the ability to find, extract, and process ARL's telemetry data in Chapter 10 files. The MATLAB code is included in Appendix A and contains more detailed information about each function.

- **derandomizer:** Derandomize PCM packet data.
- **find_first_sync:** Find the file position and relative time of the first packet of the specified channel.
- **find_last_sync:** Find the relative time of the last packet of the specified channel.

- **get_packets:** Given the position of the first packet of a specified generic channel, starting at a specified packet number, return the specified number of packets in bytes.
- **get_packets_pcm:** Given the position of the first packet of a specified PCM channel, starting at a specified packet number, return the specified number of packets in word format.
- **irig_packet2time:** Convert an IRIG-B packet to a decimal IRIG-B time.
- **irig2sfbod:** Convert an IRIG-B string to seconds from the beginning of the day.
- **make_frames:** Extract frames from derandomized PCM data and check cyclic redundancy code (CRC).

3. Example Application

The decoding library was tested on an example data file from an Irradiance Collection and Reporting System flight test, "MTS-2 AD 04252018 ROUND 2.ch10". The Telemetry Attributes Transfer Standard packet information, extracted using irig106utils,[5] is listed in Appendix B. This gives specific information about the data's recorded format. Chapter 10 data in another format are not guaranteed to be compatible with this library. The two channels of interest are channel 6, randomized PCM data, and channel 1, IRIG-B timecode data. An example script was written to decode the data, "read_ch10_data.m", listed in Appendix C.

The script extracts telemetry data from a Chapter 10 file from the beginning of the packet containing the specified start time, for the minimum number of packets, to ensure the specified duration. The main steps of the script are as follows:

1) Define the user parameters
2) Relate the IRIG time to the relative time counter
3) Find the PCM start packet and number of packets to read
4) Find the total duration of the file and the total number of packets
5) Extract the PCM packets
6) Derandomize and error check the PCM packets
7) Extract the frames from the PCM packets
8) Plot example signals

While running, the script outputs the following information:

- Chapter 10 file name
- Total duration of data in seconds and the total number of packets
- The progress searching for the first PCM packet in bytes
- An indication that the PCM packets were read from the Chapter 10 file
- The progress derandomizing the data in bits
- The total number of frames extracted and the number of failed CRC checks
- The total script runtime

The following output data were printed by the script:

```
MTS-2_FC_04262018_TM_ROUND_2.ch10
Total file time = 322.502296, npackets_max = 19703
Searching: 2000 4000 6000 8000 10000 12000 14000 16000
Reading: done!
Derandomizing... 80000 160000 240000 320000 400000 480000 560000
640000 720000 800000
Nframes=1080, Nerrors=48
Elapsed time is 7.608169 seconds.
```

Derandomizing the data takes the most execution time. It is preferable, therefore, for the derandomized PCM data to be saved in the Chapter 10 file in order to avoid this step. Figure 1 shows some example signals extracted from the telemetry data. Time zero was set to the muzzle flash time, the start of the data was set to 0.5 s before the muzzle flash, and the duration was set to 0.2 s. The total duration of displayed data was actually more than 0.2 s because the library always processes whole packets. The 48 frames that failed the CRC checks were excluded from the plots. From the quality of the plotted signals and the obvious alignment of the time base to munition launch, the correct performance of the Chapter 10 data decoding library has been demonstrated.

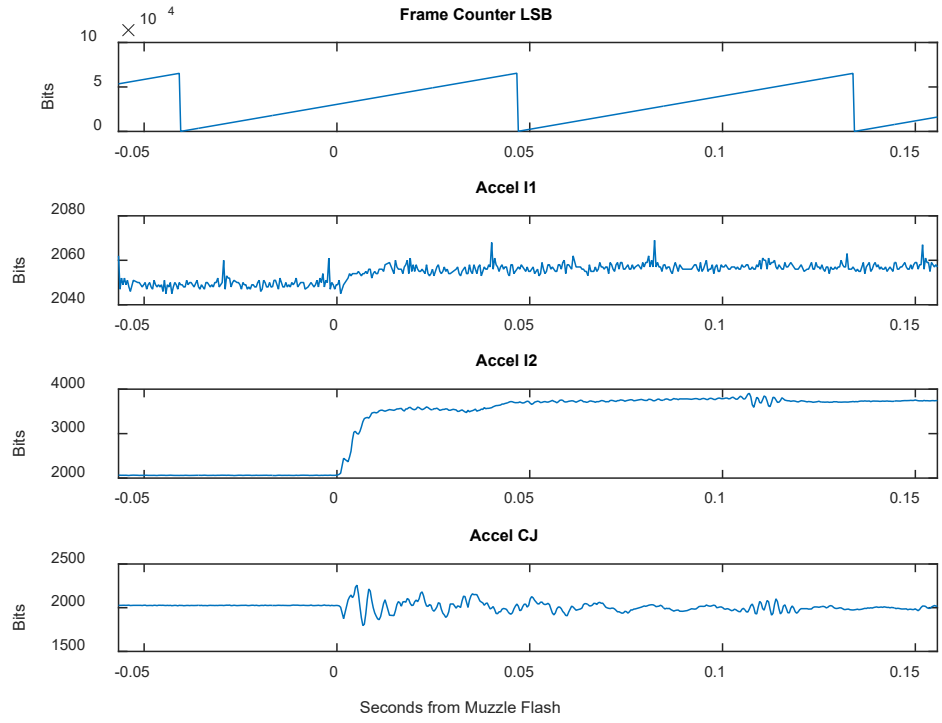

Fig. 1 Example telemetry signals extracted from a Chapter 10 file

4. Conclusion

This technical note has presented a custom library of MATLAB functions to decode IRIG Chapter 10 data, focusing on the extraction of telemetry frames and timing from PCM and IRIG-B packets. In addition, the MATLAB implementation provides a convenient platform for further data analysis. After an overview of the library, the functionality of the library was demonstrated with an example script that extracts telemetry data from a sample Chapter 10 file.

5. References

1. Don M. A low-cost software-defined telemetry receiver. Proceedings of the 51st International Telemetering Conference; 2015 Oct 26–29; Las Vegas, NV.

2. Don M. Advances in a low-cost software-defined telemetry system. Proceedings of the 53rd International Telemetering Conference; 2017 Oct 23–26; Las Vegas, NV.

3. Davis BS, Guidos BJ, Harkins TE. Complementary roles of spark range and onboard free-flight measurements for projectile development. Aberdeen Proving Ground (MD): Army Research Laboratory (US); 2009 Aug. Report No.: ARL-TR-4910.

4. Range Commanders Council Telemetry Group. Telemetry standards, IRIG standard 106-13, Part I. Chapter 10. Digital on-board recorder standard. White Sands Missile Range (NM): Range Commanders Council; 2013 June [accessed 2018 Sep]. http://www.wsmr.army.mil/RCCsite/Documents/106_Previous_Versions/106-13_Telemetry_Standards/chapter10.pdf.

5. Range Commanders Council Telemetry Group. IRIG106wiki. White Sands Missile Range (NM): Range Commanders Council [updated 2018 Nov 26; accessed 2018 Sep]. http://www.irig106.org/wiki/.

6. Telspan Data. NetView Data Fusion & Display..Concord (CA): Telspan Data; nd [accessed 2018 Sep]. https://telspandata.com/software/netview/.

7. MATLAB. Natick (MA): The Mathworks, Inc.; c1994–2018 [accessed 2018 Sep]. https://www.mathworks.com/products/matlab.html.

Appendix A. MATLAB Library Functions

This appendix appears in its original form, without editorial change.

The following MATLAB functions provide the ability to find, extract, and process the US Army Laboratory's telemetry data in Chapter 10 files.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ c ] = derandomizer( a )
%derandomizer: Derandomize PCM packet data.
% inputs:
%       a - randomized PCM bytes
% outputs
%       c - derandomized PCM bytes

n=length(a);
ab=mydec2bin(a,0);  %used to use dec2bin,  look at 1p2
clear a
reg=false(1,15);
output=false(1,n*8);
for i=1:n*8
    if mod(i,10000*8)==0
        fprintf('%d ',i)
    end
    output(i)=xor(ab(i),xor(reg(14),reg(15))); %calc output
    reg=[ab(i) reg(1:end-1)]; %clock update
end
fprintf('\n')
c=mybin2dec_byte3(output);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [bb ] = mydec2bin( b,verbose)
%% mydec2bin: converts a decimal to binary representation
% inputs
%       b - decimal bytes
%       verbose - enable bit index printing
% outputs
%       bb - binary outputs

bb=false(1,8*length(b));
for i=1:8
    if verbose
        fprintf('%d ',i);
    end
    bb(i:8:end)=bitget(b,8-i+1)==1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [a] = mybin2dec_byte3(b)
% mybin2dec_byte3: convert binary to decimal representation in bytes
% inputs
%       b - binary input
% outputs
%       a - bytes

a=zeros(1,length(b)/8,'uint8');
b=uint8(b);
for i=1:8
a=a+bitshift(b(i:8:end),8-i);
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ fpos,rtime,packet_len ] = find_first_sync( rd_len, channel ,fileID)
%find_first_sync: returns information about first packet of given channel
%   Inputs
%               rd_len - size of chunks of data to read at a time to search for
first sync, 50,000 seems to work ok
%               channel - channel number of packet you're looking for
%               fileID - ID of fileID
%       Outputs
```

```
%              fpos - file position of first packet of given channel
%              rtime - relative time counter of first packet
%              packet_len - length of first packet in bytes

sync=hex2dec(['25'; 'EB'])'; %little endian
sync=[sync typecast(uint16(channel),'uint8')];
synci=[];
fseek(fileID,0,'bof');
while isempty(synci)
    bytes=fread(fileID,rd_len,'uint8=>uint8')';
    synci=find((bytes(1:end-3)==sync(1)) & (bytes(2:end-2)==sync(2)) &
(bytes(3:end-1)==sync(3)) & (bytes(4:end)==sync(4)));
    fseek(fileID, -3, 'cof'); %go back 3 bytes, in case sync is inbetween reads
end

bytes=[bytes fread(fileID,24,'uint8=>uint8')']; %make sure have whole header
rtime=double(typecast([bytes(synci+16:synci+21) 0 0],'uint64')); %relative time of
first packet
packet_len=double(typecast(bytes(synci+4:synci+7),'uint32'));
fpos=ftell(fileID)+synci(1)+2-rd_len-24;
end




%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%function [
packet_len,rtime ] = find_last_sync( rd_len, channel,fileID )
%find_last_sync: returns information about first packet of given channel
%   Inputs
%              rd_len - size of chunks of data to read at a time to search for
first sync, 50,000 seems to work ok
%              channel - channel number of packet you're looking for%%
%              fileID - ID of fileID
%       Outputs
%              rtime - relative time counter of first packet
%              packet_len - length of first packet in bytes

sync=hex2dec(['25'; 'EB'])'; %little endian
sync=[sync typecast(uint16(channel),'uint8')];
i=1;
synci=[];
while isempty(synci)
    if i==1
        fseek(fileID, -rd_len*i, 'eof');
    else
        fseek(fileID, -rd_len*i+3, 'eof'); %go back 3 bytes, in case sync is
inbetween reads
    end
    bytes=fread(fileID,rd_len,'uint8=>uint8')';
    synci=find((bytes(1:end-3)==sync(1)) & (bytes(2:end-2)==sync(2)) &
(bytes(3:end-1)==sync(3)) & (bytes(4:end)==sync(4)));
    i=i+1;
end
if i>2
    bytes=[bytes fread(fileID,24,'uint8=>uint8')']; %make sure have whole header
end
packet_len=double(typecast(bytes(synci+4:synci+7),'uint32'));
rtime=double(typecast([bytes(synci+16:synci+21) 0 0],'uint64'));
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ analog ] =
get_packets(sub_samp,fpos,ChanID_get,packet_len,header_size,npackets,fileID,nstart
,signed,output_en)
% get_packets_pcm: gets pcm data from multiple packets in bytes
%   Inputs
%              sub_sample - amount to subsample data
%              fpos - file position of first packet
```

```
%               ChanID_get - ID of channel to get
%               packet_len - length of packets in bytes
%               header_size - length of header in bytes
%               npackets - number of packets to get
%               fileID - ID of file
%               nstart - packet to start from
%               signed - string indicating the output type
%               output_en - enable verbose runtime output_en
%       Outputs
%               analog - vector of packet data
%


fseek(fileID, fpos, 'bof');
analog=zeros(1,npackets*(packet_len-header_size)/sub_samp,signed);
for di=1:(npackets+nstart-1)
    if mod(di,500)==0 && output_en
        fprintf('%d ',di);
    end
    if di>=nstart
        bytes=fread(fileID,packet_len,'uint8=>uint8')'; %read data, now file pos
at next packet
        tmp=typecast(bytes(29:end),signed);
        starti=(di-nstart)*(packet_len-header_size)/sub_samp+1;
        endi=(packet_len-header_size)*(di-nstart+1)/sub_samp;
        analog(starti:endi)=tmp(1:sub_samp:end);
    else
        fseek(fileID,packet_len,'cof');
    end


    %now find next header, find packet length - keep going till next 0002 ChanID
    ChanID=-1;
    while ChanID~=ChanID_get
        bytes=fread(fileID,24,'uint8=>uint8')'; %read header
        ChanID=typecast(bytes(3:4),'uint16');
        packet_len=typecast(bytes(5:8),'uint32');
        fseek(fileID, packet_len-24, 'cof'); %goto next packet
    end
    fseek(fileID, -1*double(packet_len), 'cof'); %go back to prev packet
end
if output_en
    fprintf('Done!\n');
end
end



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ pcm,timea ] =
get_packets_pcm(sub_samp,fpos,ChanID_get,packet_len,header_size,npackets,fileID,ns
tart,output_en)
%get_packets_pcm: gets pcm data from multiple packets in bytes
%   Inputs
%               sub_sample
%               fpos - file position of first packet
%               ChanID_get - ID of channel to get
%               packet_len - length of packets in bytes
%               header_size - length of header in bytes
%               npackets - number of packets to get
%               fileID - ID of file
%               nstart - packet to start from
%               output_en - enable verbose runtime output_en
%       Outputs
%               pcm - vector of packet data in bytes
%               timea - array of relative time stamps of all packets (double)

fseek(fileID, fpos, 'bof'); %goto beginning of 1st packet
if output_en
    fprintf('Searching: ');
end
```

```
pcm=zeros(1,npackets*(packet_len-header_size)/sub_samp/2,'uint16'); %preallocate
data
timea=zeros(1,npackets);
for di=1:(npackets+nstart-1)
    if (mod(di,2000)==0 && output_en)
        fprintf('%d ',di);
    end
    if di>=nstart %only save data from packet nstart
        if (di==nstart && output_en)
            fprintf('\nReading: ');
        end
        bytes=fread(fileID,header_size,'uint8=>uint8')'; %read header
        rtime=double(typecast([bytes(17:22) 0 0],'uint64')); %relative time of
first packet
        bytes=fread(fileID,packet_len/2-header_size/2,'uint16=>uint16')'; %read
data, now file pos at next packet
        starti=(di-nstart)*(packet_len/2-header_size/2)/sub_samp+1;
        endi=(packet_len/2-header_size/2)*(di-nstart+1)/sub_samp;
        pcm(starti:endi)=bytes(1:sub_samp:end); %save data in pcm array
        timea(di-nstart+1)=rtime;
    else
        fseek(fileID,packet_len,'cof'); %if not at packet nstart yet, skip
    end

    %now find next header, find packet length - keep going till next 0002 ChanID
    ChanID=-1;
    while ChanID~=ChanID_get
        bytes=fread(fileID,header_size,'uint8=>uint8')'; %read header
        ChanID=typecast(bytes(3:4),'uint16');
        packet_len=double(typecast(bytes(5:8),'uint32'));
        fseek(fileID, packet_len-header_size, 'cof'); %goto next packet
    end
    fseek(fileID, -1*double(packet_len), 'cof'); %go back to prev packet
end
if output_en
    fprintf('done!\n');
end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ time ] = irig_packet2time( bytes )
%irig_packet2time: convert Chapter 10 irig-b packet in bytes to time d:h:m:s
%   inputs
%              bytes - Chapter 10 irig-b packet in bytes
%       outputs
%              time - time structure

tmp=dec2hex(bytes);
time.s=str2double(tmp(2,:));
time.m=str2double(tmp(3,:));
time.h=str2double(tmp(4,:));
time.d=str2double([tmp(6,:) tmp(5,:)]);
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [ seconds ] = irig2sfbod( string )
%irig2sfbod: Convert an IRIG-B string to seconds from the beginning of the day.
% inputs:
%       string - IRIG-B string
% outputs:
%       seconds - seconds from the beginning of the day

    start_time=sscanf(string,'%f:%f:%f:%f');
    seconds=start_time(2)*3600+start_time(3)*60+start_time(4);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function
[frames16,ftime,errors]=make_frames(a,irig_time_data_start,byte_dt,frame_bytes,syn
c,do_crc)
```

```
%   inputs
%               a - PCM byte array
%               irig_time_data_start - start time
%               byte_dt - the time duration of one byte
%               frame_bytes - number of bytes in a frame
%               sync - telemetry sync word
%       outputs
%               frames16 - frame matrix in uint16 format
%               ftime - frame timestamps
%               errors - CRC checking results

width=8;
bit_dt=byte_dt/8;
frame_len=frame_bytes/2;
max_frames=ceil(length(a)/frame_bytes);
synci=zeros(width,max_frames);
frames=zeros(frame_bytes,max_frames,'uint8');
ftime=zeros(1,max_frames);
fi=1; %frame index
for shift=0:width-1
    am=bitor(bitshift(a(1:end-1),shift),bitshift(a(2:end),shift-width));
    tmp=find((am(1:end-3)==sync(1)) & (am(2:end-2)==sync(2)) & (am(3:end-
1)==sync(3)) & (am(4:end)==sync(4)));
    for i=1:length(tmp)
        if tmp(i)<(length(am)-frame_bytes-1)
            frames(:,fi)=am(tmp(i):tmp(i)+frame_bytes-1);
            ftime(fi)=irig_time_data_start+(tmp(i)-1)*byte_dt+shift*bit_dt;
            fi=fi+1;
        end
    end
    synci(shift+1,1:length(tmp))=tmp;
end
clear a am
% debug, view vector of bins: reshape(dec2bin(am(8,:))',1,length(am(1,:))*8)

frames=frames(:,1:fi-1);
ftime=ftime(:,1:fi-1);
[ftime,ind]=sort(ftime); %when doing one row of shifti at a time, things can be
out of order, now reorder
frames=frames(:,ind);
nframes=length(frames(1,:));
frames16=zeros(frame_len,nframes,'uint16');
for i=1:nframes
    frames16(:,i)=swapbytes(typecast(frames(:,i),'uint16'));
end
clear frames

if do_crc
    csum=sum(frames16(1:frame_len-1,:)); %sum changes it to double
    csum=uint16(bitand(uint32(hex2dec('0000FFFF')),uint32(csum)));
    csum(csum>1)=bitcmp(csum(csum>1)-1)+1;  %crc - all words added, then crc=0-
sum+1;
    csum(csum==1)=0;
    csum(csum==0)=1;
    errors=(csum~=uint16(frames16(frame_len,:)));
else
    errors=false(1,nframes);
end

end
```

Appendix B. Telemetry Attributes Transfer Standard Information of an Example Irradiance Collection and Reporting System Chapter 10 File

This appendix appears in its original form, without editorial change.

This is the Telemetry Attributes Transfer Standard (TMATS) information for the example Irradiance Collection and Reporting System (ICRS) Chapter 10 file that was decoded and displayed in Fig. 1 in the main report.

```
================================================================================================
Chapter 10 Filename:C:\Users\Administrator\Documents\Work\ICRS\MTS-2 AD 04252018 ROUND 2.ch10
TMATS Packet Info:
Program Name            :_
Number of Channels      :17
Channel   DataType     Enable    BitRate   WordLen   NumFrames Words/Frame Bits/Frame  DataMode
-------   --------     ------    -------   -------   --------- ----------- ----------  --------
1   IRIG           Enabled
2   ANALOG         Enabled
3   ANALOG         Disabled
4   ANALOG         Disabled
5   ANALOG         Disabled
6   PCM            Enabled    4000000       16          1           32        528  ThroughPut
7   PCM            Disabled    512000       16          1           32       2047  ThroughPut
8   PCM            Disabled   4000000       16          1           31        512  ThroughPut
9   PCM            Disabled   1000000       16          1           31        512  ThroughPut
10   ANALOG         Disabled
11   ANALOG         Disabled
12   ANALOG         Disabled
13   ANALOG         Disabled
14   PCM            Disabled   1000000       16          1           31        512  ThroughPut
15   PCM            Disabled   1000000       16          1           31        512  ThroughPut
16   PCM            Disabled   1000000       16          1           31        512  ThroughPut
17   PCM            Disabled   1000000       16          1           31        512  ThroughPut
================================================================================================
Offset      :  0x0000000000000000
ChanID      :  0       Sync        :  0xEB25  PktLen      :  12800
DataLen     :  11528   Ver         :  4       Seq         :  0
REFTIME ERR :  0       OVERFLOW ERR:  0       CHKSUM IND  :  NO CHK
SECOND HDR  :  NO      TIME SOURCE :  PktHdr 48-Bit Ref Time
Type        :  1 SETUP TimeLo      :  0xFCBE  TimeMid     :  0x94AB
TimeHi      :  0x0034  ChkSum      :  56526

Packet's Data Efficiency : 90.0625
Packet's Relative Time Counter : 225832598718
SETUP CHAN SPEC: 0x00000008
IRIG 106 Chapter 10 Version: 0x08 (IRIG-106-09)
TMATS Format: 0 ASCII     Setup Record Conf Change: 0 NO

COMMENT: Original Recording File - MTS-2 AD 04252018 ROUND 2.ch10;
COMMENT: RSN Name - File0011;
COMMENT: ----------- Channel Summary -------------;
COMMENT: 17 channels (3 enabled);
COMMENT:  Id   Data Type  Data Source Id;
COMMENT: ----  ---------  ---------------;
COMMENT:   1   Time       TIME01; COMMENT:   2   Analog     IRIG B;
COMMENT:   6   PCM        RF-1 J-104;
```

Appendix C. Modulation Simulation MATLAB Code

This appendix appears in its original form, without editorial change.

This example MATLAB script uses the library functions to decode an example Irradiance Collection and Reporting System (ICRS) flight test Chapter 10 file. The results are displayed in Fig. 1 of the main report.

```
clear
close all

%% User parameters
% Assumes 16 bit words, crc is last word
% Gets data from beginning of packet with start_time, till last packet with
start_time+duration
% File time is adjusted if it falls outside of the file boundaries
% Need to read at least 2 packets (duration > 0.01636 s)

load_from_mat_file=0; %load saved frames from mat file
do_save=0; %save frames
save_suffix='launch'; %append to end of saved file name
duration=0.2; %in seconds
shift_start_time=0; %seconds to shift start time
irig_channel=1;
pcm_channel=6;
derandomize=1; %derandomization enable
do_crc=1; % CRC error deterction enable
remove_errors=1; % remove detected errors
sync=hex2dec(['FE'; '6B'; '28'; '40'])';
frame_len=48; %words in frame

dirc{1} = 'C:\Users\Administrator\Documents\Work\Customer\RITA\ICRS\'; %need '\'
at end

filenamec{1} = 'MTS-2_FC_04262018_TM_ROUND_2_trim.ch10';  %good data!!!!!!

start_timec{1}='116:18:03:04.268'; % 'Day:Hour:Minutes:Seconds'
zero_timec{1}= '116:18:03:04.318';


for filei=1:length(dirc)

    %% Parameters
    tic
    filename=filenamec{filei};
    fprintf('\n%s\n',filename)
    zero_time=zero_timec{filei};
    start_time=start_timec{filei};
    dir=dirc{filei};
    start_seconds=irig2sfbod(start_time)+shift_start_time;  %seconds of beginning
of data (ignores days)
    zero_seconds=irig2sfbod(zero_time);  %seconds to consider as zero for plotting
(ignores days)

    if ~load_from_mat_file
        %% Program constants, computed parameters
        frame_bytes=frame_len*2;
        rd_len=50000;
        header_size=28; %bytes at beginning of packet before data starts (seemed
like should start 4 bytes earlier???)
        full_filename=[dir filename];

        %% Relate IRIG time to relative time counter
        fileID = fopen(full_filename,'r');
        [fpos,rtime_irig,packet_len] = find_first_sync(rd_len, irig_channel,
fileID); %find first irig sync
        time =
irig_packet2time(get_packets(1,fpos,irig_channel,packet_len,header_size,1,fileID,1
,'uint8',0)); %get time
        seconds_fbod=time.h*3600+time.m*60+time.s; %seconds from beginning of day
of first time packet
```

```
        %% Find start packet and npackets
        [ fpos,rtime,packet_len] = find_first_sync(rd_len, pcm_channel, fileID);
%find first pcm sync
        [~, rtimea] =
get_packets_pcm(1,fpos,pcm_channel,packet_len,header_size,2,fileID,1,0);
        packet_duration=diff(rtimea)/10e6;
        irig_time_pcm=seconds_fbod+(rtime-rtime_irig)/10e6; %irig time of
beginning of pcm data
        npackets=ceil(duration/packet_duration);
        nstart=floor((start_seconds-irig_time_pcm)/packet_duration)+1;

        %% estimate total time and npackets in file
        [~,rtime_end] = find_last_sync(rd_len, pcm_channel, fileID); %find last
packet
        total_time=(rtime_end-rtime)/10e6;
        npackets_max=floor(total_time/packet_duration);
        fprintf('Total file time = %f, npackets_max =
%d\n',total_time,npackets_max);
        if nstart<1
            nstart=1;
            fprintf('Start time delayed to %s!!!\n',sfbod2irig(irig_time_pcm));
        end
        if (nstart+npackets-1)>npackets_max
            npackets=npackets_max-nstart+1;
            fprintf('Duration clipped to %f s!!!\n',npackets*packet_duration);
        end

        %% get packets
        [a, rtimea] =
get_packets_pcm(1,fpos,pcm_channel,packet_len,header_size,npackets,fileID,nstart,1
); %get first 2 packets
        packet_dt=(rtimea(end)-rtimea(1))/(npackets-1)/10e6;
        byte_dt=packet_dt/(packet_len-header_size);

        irig_time_data_start=seconds_fbod+(rtimea(1)-rtime_irig)/10e6; %irig time
of beginning of pcm data

        %% Derandomize
        fprintf('Derandomizing... ')
        a=typecast(swapbytes(a),'uint8');
        if derandomize
            a=derandomizer(a);
        end

        %% Do bit sync, make frames and time array, get crc errors

[frames16,ftime,errors]=make_frames(a,irig_time_data_start,byte_dt,frame_bytes,syn
c,do_crc);
        clear a
        nframes=length(ftime);
        nerrors=sum(errors);
        fprintf('Nframes=%d, Nerrors=%d\n',nframes,nerrors);
        toc

    else
        tmp=textscan(filenamec{filei},'%s','Delimiter','.');
        load(sprintf('%s_%s.mat',char(tmp{1}(1)),save_suffix));
    end

    %% Plotting
    if remove_errors
        midasi=(frames16(3,:)==0) & ~errors;
    else
        midasi=(frames16(3,:)==0);
    end
    mtime=ftime(midasi)-zero_seconds;

    if length(mtime)>2
    figure
    subplot(4,1,1)
    plot(mtime,frames16(46,midasi))
```

```
    xlim([mtime(1) mtime(end)]);
    title('Frame Counter LSB')
    ylabel('Bits')
    subplot(4,1,2)
    plot(mtime,frames16(7,midasi))
    xlim([mtime(1) mtime(end)]);
    title('Accel I1')
    ylabel('Bits')
    subplot(4,1,3)
    plot(mtime,frames16(11,midasi))
    xlim([mtime(1) mtime(end)]);
    title('Accel I2')
    ylabel('Bits')
    subplot(4,1,4)
    plot(mtime,frames16(13,midasi))
    xlim([mtime(1) mtime(end)]);
    title('Accel CJ')
    xlabel('Seconds from Muzzle Flash')
    ylabel('Bits')
    else
        fprintf('Not plotting!!!  Less than 2 midas frames.\n')
    end

    if do_save
        tmp=textscan(filenamec{filei},'%s','Delimiter','.');
        save(sprintf('%s_%s.mat',char(tmp{1}(1)),save_suffix), 'errors',
'frames16', 'ftime')
    end
end
```

List of Symbols, Abbreviations, and Acronyms

ARL	US Army Research Laboratory
CRC	cyclic redundancy code
ICRS	Irradiance Collection and Reporting System
IRIG	Inter-Range Instrumentation Group
PCM	pulse code modulated
TMATS	Telemetry Attributes Transfer Standard

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
23 (PDF)	ARL RDRL WML F M DON B ALLIK B J ACKER T G BROWN S BUGGS E BUKOWSKI J COLLINS J CONDON B DAVIS D EVERSON R HALL J HALLAMEYER M HAMAOUI T HARKINS M ILG B KLINE J MALEY C MILLER B NELSON D PETRICK K PUGH N SCHOMER B TOPPER