Application Note

MATLAB Application

MG3700A
Vector Signal Generator

Anritsu

Slide 1

Application Note

- MATLAB® Application -

Anritsu

MG3700A

Vector Signal Generator

January 2008
(3.00)

Discover What's Possible™
MG3700A-E-F-11

Slide 1

Anritsu

Slide 2

Contents

Discover What's Possible™
MG3700A-E-F-11

Slide 2

Anritsu

1

Introduction

- Electronic Design Automation (EDA) tools, such as MATLAB and Microwave Office, can save IQ simulation data to CSV text files.
  - » Easy comparison between simulation and measured data
- The IQproducer application software (MG3700A standard accessory) can import user I/Q sample data from CSV files into MG3700A, making it useful for various applications.
  - » R&D into next-generation wireless systems, RFID, PWM, analog modulation such as AM, FM, and PM

User-specific Software

Processing (Simulating) signals using C program or EDA tools such as MATLAB

Creating CSV files of baseband signals

Converting to signal pattern files for MG3700A

FTP

IQproducer for MG3700
File System Transfer & Setting Simulation File Con Help
Convert
AWGN
Clipping

2 110 000 000.00
0.00 dBm

Anritsu
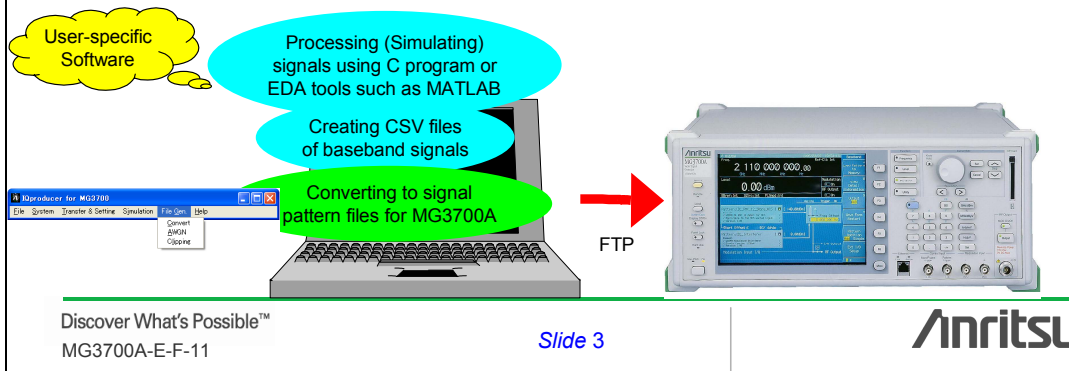
Introduction

- MG3700A supports Arbitrary Waveform Generator (AWG) features
  - » Sample rate (Fs)
    - – 20 k to 160 MHz
  - » Waveform memory
    - – 2× 128 Msample/channel (1 GB)
    - – 2× 256 Msample/channel (2 GB) * with Option Memory upgrade
      - • Dual baseband memory
  - » DAC resolution
    - – 14 bits
  - » Marker output
    - – 3 definable TTL

- This document presents ⚛ MATLAB usage examples.

Anritsu

2

Modulation Basics

- Modulation imposes source data on a passband signal with carrier frequency f_c by using amplitude and/or phase perturbations.

Modulating signal $m(t)$ — Baseband circuits — Baseband signal $g(t)$ — RF circuits — Passband signal $s(t)$
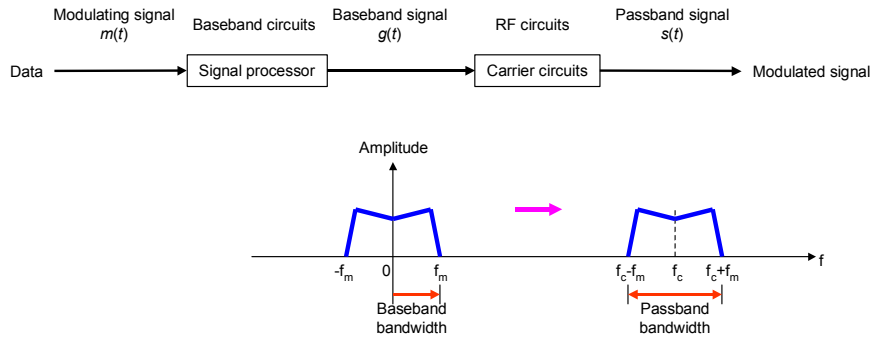
Data → Signal processor → Carrier circuits → Modulated signal

Amplitude

$-f_m$ 0 f_m f_c-f_m f_c f_c+f_m f

Baseband bandwidth   Passband bandwidth

Modulation Transmitter

- AM-PM technique

Baseband circuits   RF circuits

$m(t)$ → Baseband signal processor

$R(t)$

f_c $\cos(\omega_c t)$

$\theta(t)$ → ϕM $\cos[\omega_c t + \theta(t)]$

$v(t) = R(t)\cos[\omega_c t + \theta(t)]$

Equivalent
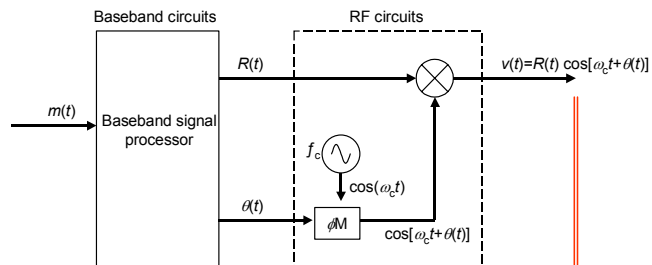
$$R(t) = \sqrt{x^2(t) + y^2(t)}$$

$x(t) = R(t)\cos\theta(t)$
$y(t) = R(t)\sin\theta(t)$

- Quadrature (IQ) technique

Baseband circuits   RF circuits

$m(t)$ → Baseband signal processor

$x(t)$
I

f_c $\cos(\omega_c t)$

$-\pi/2$ shift $\sin(\omega_c t)$

$y(t)$
Q

Σ + −

$v(t) = x(t)\cos(\omega_c t) - y(t)\sin(\omega_c t)$

$\omega_c = 2\pi f_c$
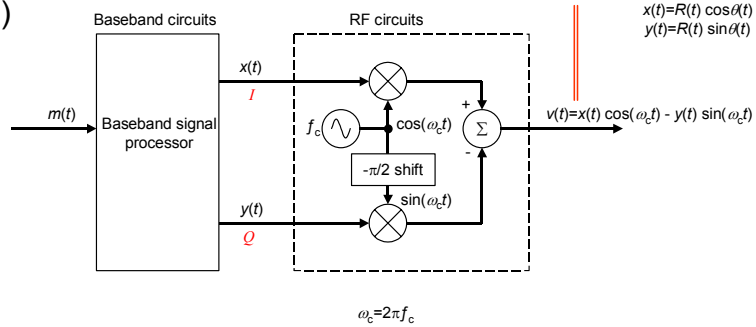
Baseband Waveform

Modulation scheme	AM-PM		IQ modulation		Remark
	$R(t)$	$\theta(t)$	$x(t)$	$y(t)$	
AM	$A_c[1+m(t)]$	0	$A_c[1+m(t)]$	0	$m(t)=\mu\sin(2\pi f_m t)$ μ: Modulation depth f_m: Modulation freq.
PM	A_c	$D_p m(t)$	$A_c\cos[D_p m(t)]$	$A_c\sin[D_p m(t)]$	D_p: Phase deviation [rad/V]
FM	A_c	$D_f\int_{-\infty}^{t} m(\sigma)d\sigma$	$A_c\cos\left[D_f\int_{-\infty}^{t} m(\sigma)d\sigma\right]$	$A_c\sin\left[D_f\int_{-\infty}^{t} m(\sigma)d\sigma\right]$	D_f: Frequency deviation [rad/V·s]
QM	$A_c\sqrt{m_1^{2}(t)+m_2^{2}(t)}$	$\tan^{-1}\dfrac{m_2(t)}{m_1(t)}$	$A_c m_1(t)$	$A_c m_2(t)$	$m_1(t)=\pm 1$ $m_2(t)=\pm 1$

QM: Quadrature Modulation

/Inritsu

Baseband Signal

- Any type of modulated signal may be generated using either the AM-PM technique or quadrature technique.
- Any modulation scheme can be achieved by selecting the appropriate software algorithm.
- In a Vector Signal Generator (VSG), the passband signal $s(t)$ is often partitioned into two channels, one for $x(t)$ called the I (in-phase) channel and one for $y(t)$ called the Q (quadrature-phase) channel.
- In digital computer simulations of passband signals, the sampling rate used in the simulation can be minimized by working with the baseband signal, instead of with the passband signal $s(t)$, because the baseband signal is equivalent of the passband signal.
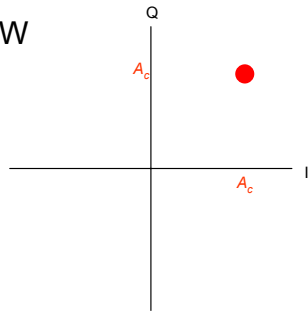
/Inritsu

Fundamental I/Q Constellation

- CW

- AM

Magnitude $R(t)$

- FM

Angular velocity [rad/s]

- PSK

Phase shift $\pi/2$ [rad]

A_c $-A_c$ A_c $-A_c$
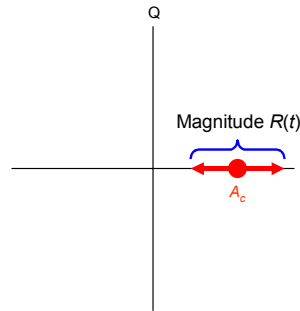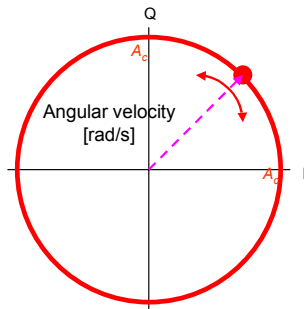
Understanding Signal Pattern File

- Signal pattern files consist of three data:
  - » IQ data
  - » Marker data
  - » File header

User-created *I/Q data* and *Marker data*

Text file to binary file
Large <<< File size >>> Small

I/Q data scaled to DAC input values.

.wvd
Binary file

File header

.wvi
Text file

FTP connection

Download →
← Extract

HDD

FTP

CF

I/Q Data
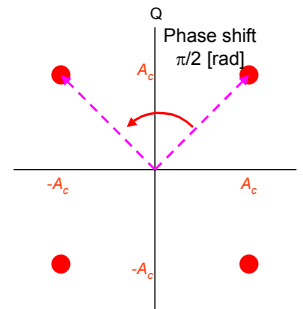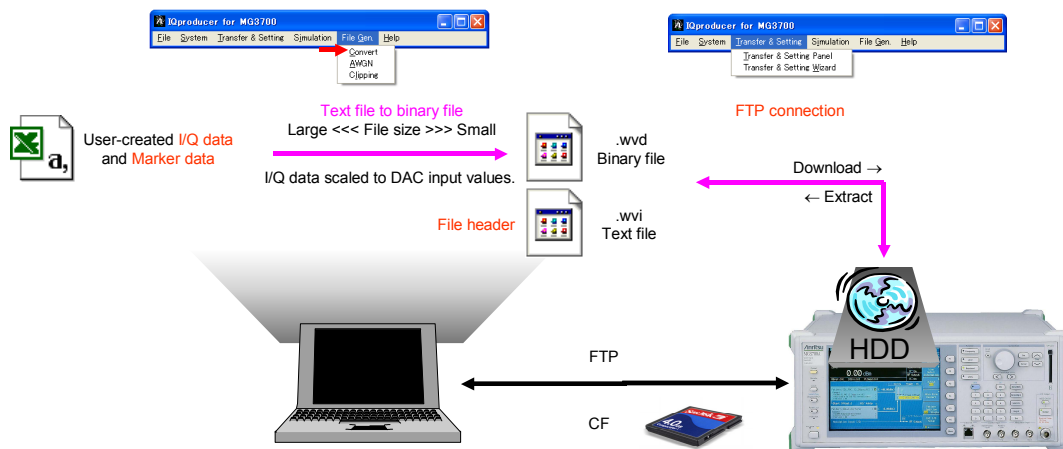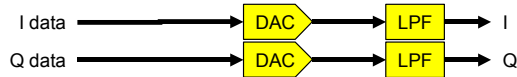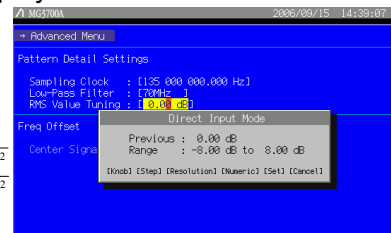
- The data file (.wvd) contains the I and Q data points (signed 14-bit integers for each I and Q data point). Each I/Q point equals one baseband waveform point.
- The MG3700A uses a 14-bit DAC to process each of the 2-byte integer values for the I and Q data points. The 14-bit DAC has a range of 0 to 16383, but MG3700A divides this range between positive and negative values.

I/Q data range	DAC range	I/Q output voltage	
– 8191	16383	V_{max}	Positive (+) full scale
– 0	8191	0 V	
– -8192	0	V_{min}	Negative (-) full scale

I/Q Data Range

- The MG3700A performance is guaranteed within I/Q DAC RMS value 1157 to 1634.
  - » Tunable DAC RMS value considered crest factor (peak power/RMS power)
  - » Tunable $\sqrt{I^2+Q^2}$ RMS power on MG3700A display

$78.27\,dB \approx 20\log_{10}\sqrt{8191^2}$

$67.28\,dB \approx 20\log_{10}\sqrt{1634^2+1634^2}$

$64.28\,dB \approx 20\log_{10}\sqrt{1157^2+1157^2}$

6
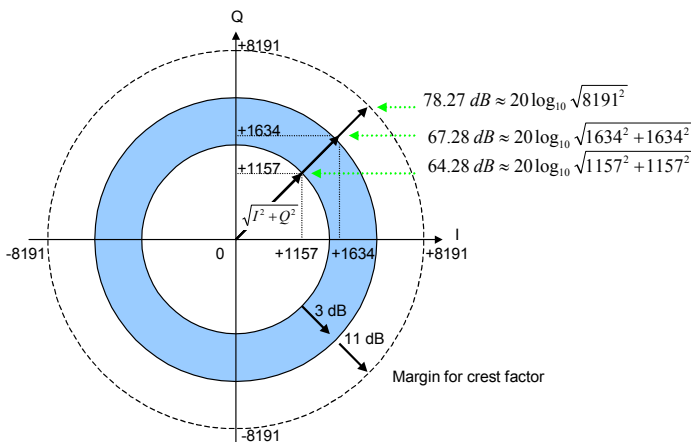
Marker Data

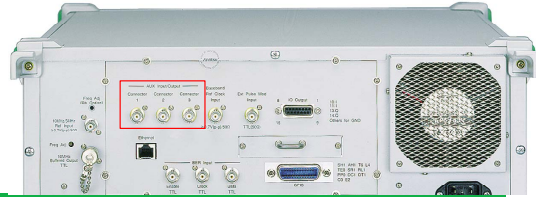- The marker data uses 4 bits per I/Q data point to set the state of the four markers either On (1) or Off (0) for each I/Q data point.
- The marker data consists of three event markers and an RF gate flag.
- When an event marker is active (On), it outputs a trigger signal at the rear-panel marker connector corresponding to active marker number 1 to 3.
- An RF gate flag is used for pulse modulation and drives the internal pulse modulator.

  » When I/Q data is imported without marker data, event markers are set automatically to inactive (Off), and the RF gate flag is set to active (On).
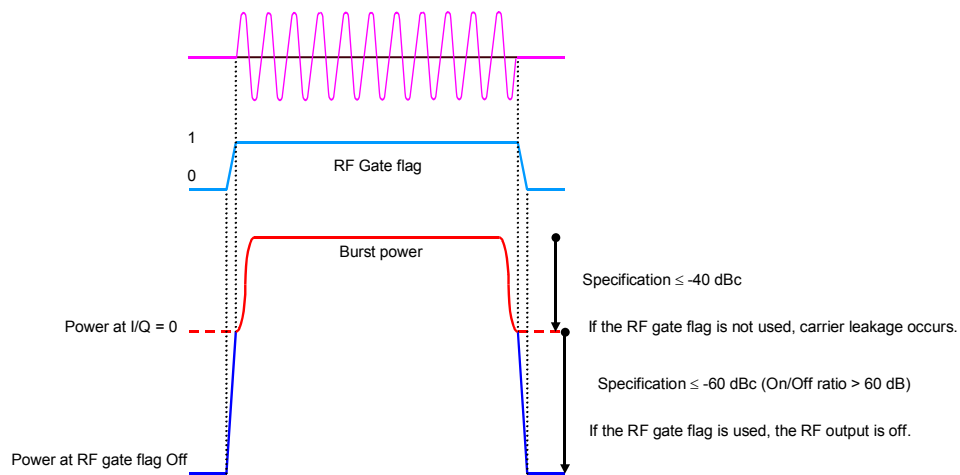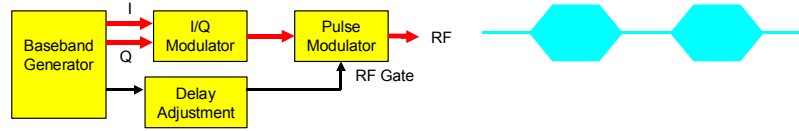
Internal Pulse Modulator

Specification ≤ -40 dBc

If the RF gate flag is not used, carrier leakage occurs.

Specification ≤ -60 dBc (On/Off ratio > 60 dB)

If the RF gate flag is used, the RF output is off.

Timing

- Because markers are set at each I/Q data point, the marker data contains the same number of samples as I/Q data points.
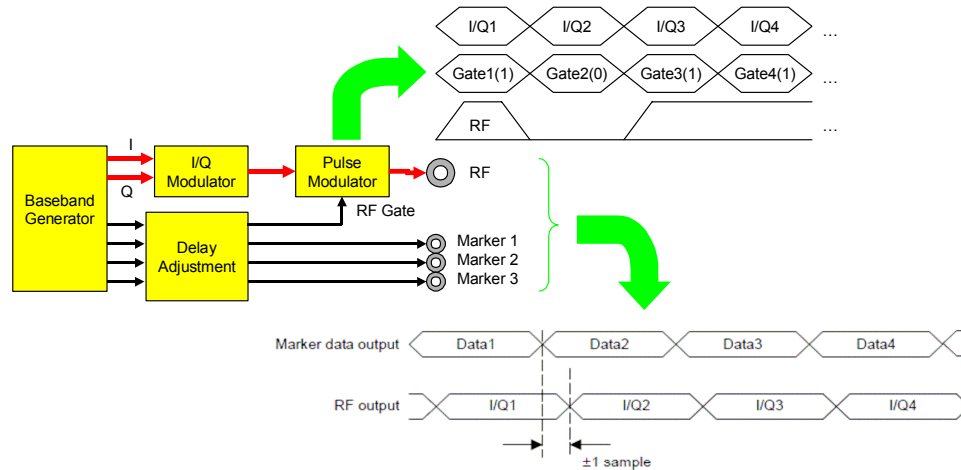
- The time difference between the event markers output connector and RF connector is adjusted to be within ±1 sample.

File Header

- The data file (.wvd) and file header (.wvi) have the same file name. The MG3700A stores the files in same package folder.
- The file header contains settings for the ARB modulation format, such as sample rate, number of samples, I/Q DAC RMS value, etc.
  - » When the MG3700A finds unspecified header settings, it either uses the default settings, or the settings from previous signal pattern if a signal pattern was played previously.

Data File Size

- The data file (.wvd) is loaded into baseband memory, meaning that the occupied memory size equals the data file size.
- A baseband waveform consists of samples. One sample contains 4 bytes.
  - » I/Q Data + Marker Data = 1 Waveform Sample
    - 14 bits I   4 bits   4 bytes (32 bits)
    - 14 bits Q

Channel
I , Q (, Marker 1, Marker 2, Marker 3, RF gate flag)

Sampling rate (sample per second) Fs:
    Sampling frequency expressed in Hz
Sampling interval Ts=1/Fs:
    Time between samples in uniform sampling

Maximum of
256,000,000 sample
512,000,000 sample * with Option

Programming Examples to Create I/Q Data File

- There are various programming environments to create ARB I/Q data.

  Generally there are two types:
  - » Simulation software (EDA tool)
    - – MATLAB, Microwave Office, etc.
  - » Advanced programming languages
    - – C/C++, VB, LabView, etc.

- The example uses MATLAB, and shows the programmable M-files.
  - M-file: MATLAB program for command script

Analog Modulation

- Analog modulation schemes are most basic techniques and it is simple to create I/Q data.

    Basic analog modulation schemes:
    » Amplitude Modulation (AM)
    » Frequency Modulation (FM)
    » Phase Modulation (PM)
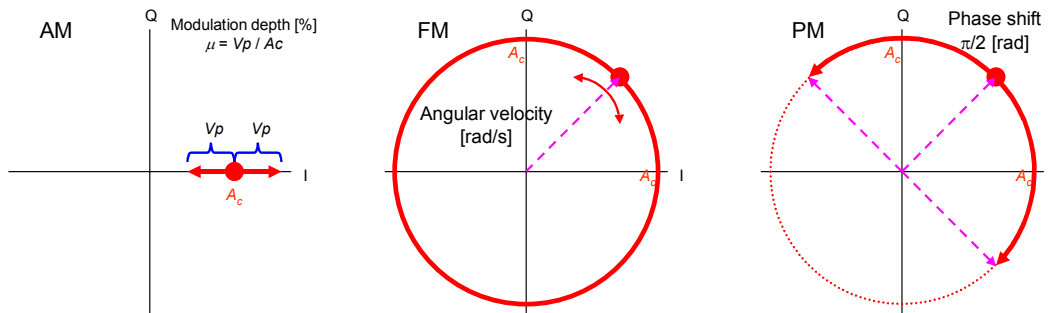
AM
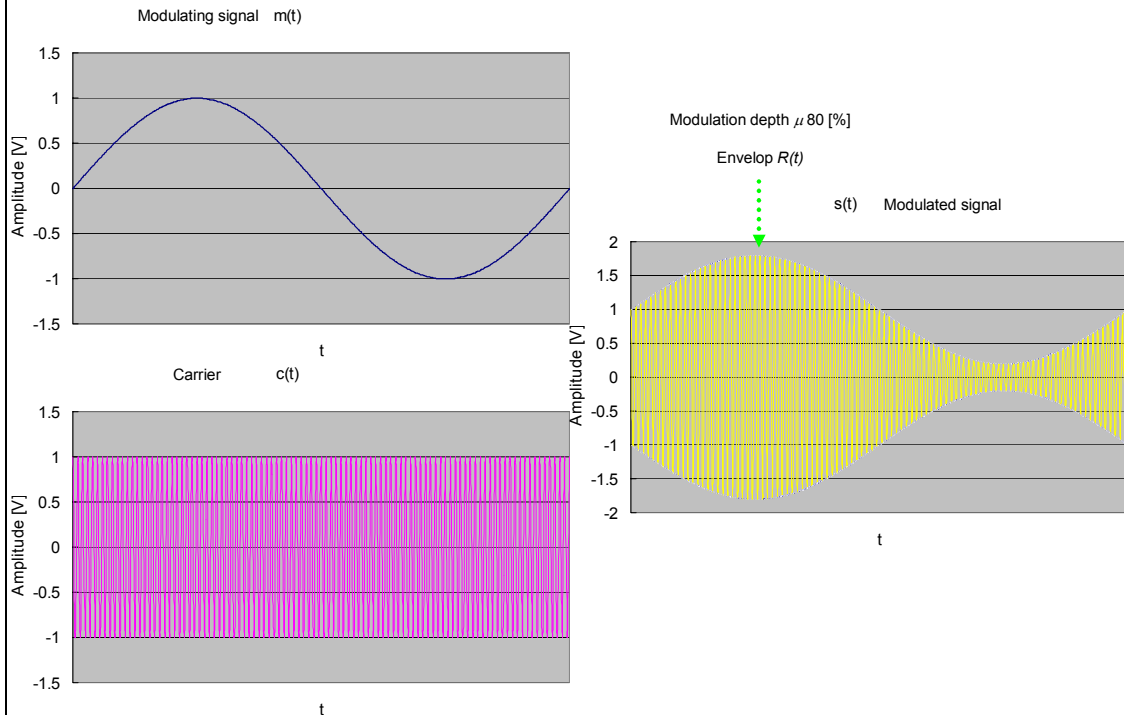
AM   Programming Example

- I = $x(t) = Ac[1+m(t)] = 1+\mu \sin(2\pi f_m t)$
- Q = $y(t) = 0$
  - *Ac*: Carrier signal amplitude = 1
  - μ: Modulation depth = Modulating signal peak/Carrier signal peak (amplitude)

```
clear all, close all, clc

fm = 1;                  % Modulating frequency [kHz]
Depth = 0.8;             % Modulating depth
CSVfile = 'AM1kHzDepth80%.csv'

DataPoints = 1000;
Fs = fm*DataPoints       % Sampling rate [kHz]

t = 0:DataPoints-1;
I = 1+Depth*sin(2*pi*fm*t/DataPoints);
Q = zeros(1,DataPoints);
data = [I' Q'];
csvwrite(CSVfile,data);
```

FM

Modulating signal   m(t)

Df

Peak frequency deviation Δf 50 [kHz]

Carrier   c(t)

s(t)   Modulated signal
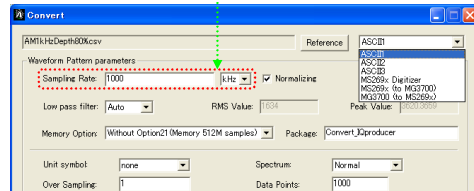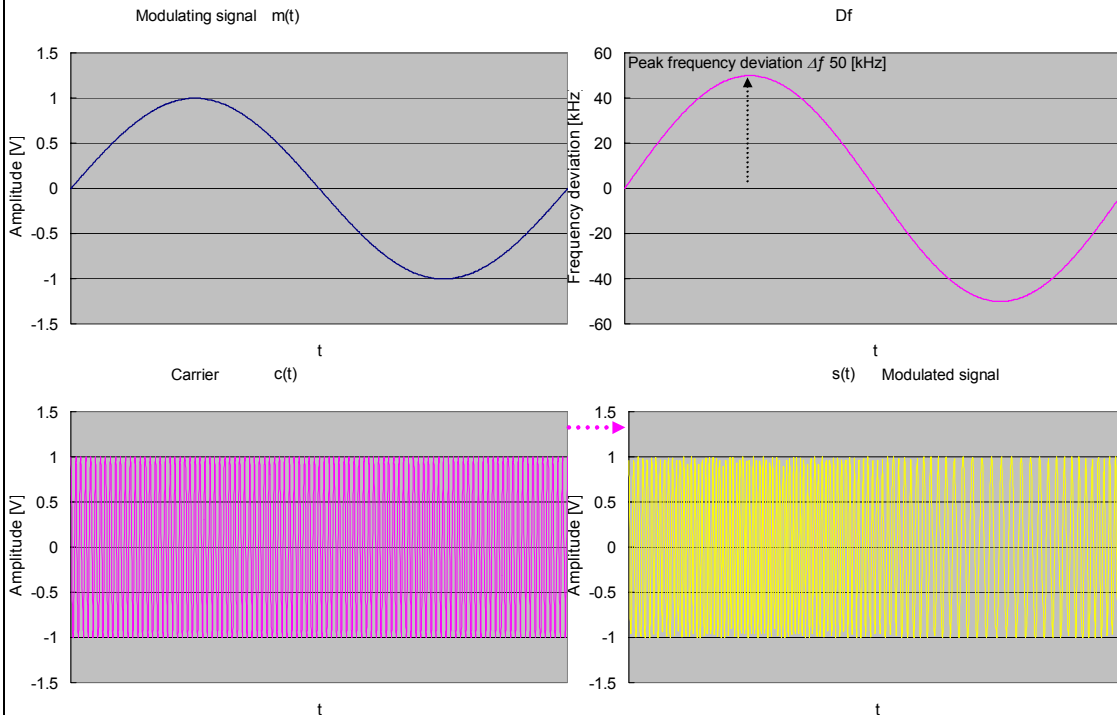
FM   Programming Example

- I = $x(t) = A_c \cos\left[D_f \int_{-\infty}^{t} m(\sigma)d\sigma\right]$ $= \cos[\beta \sin(2\pi f_m t)]$
- Q = $y(t) = A_c \sin\left[D_f \int_{-\infty}^{t} m(\sigma)d\sigma\right]$ $= \sin[\beta \sin(2\pi f_m t)]$
  - A_c: Carrier signal amplitude = 1
  - β: Modulation index = $\Delta f/f_m$
  - Δf: Peak frequency deviation

```
clear all, close all, clc

fm = 1;                    % Modulating frequency [kHz]
Deviation = 50;            % Peak frequency deviation [kHz]
CSVfile = 'FM1kHzDeviation50kHz.csv'

DataPoints = 1000;
Fs = fm*DataPoints         % Sampling rate [kHz]

t = 0:DataPoints-1;
I = cos(Deviation/fm*sin(2*pi*fm*t/DataPoints));
Q = sin(Deviation/fm*sin(2*pi*fm*t/DataPoints));
data = [I' Q'];
csvwrite(CSVfile,data);
```

PM

Modulating signal   m(t)

Dp

Peak phase deviation 5 [rad] ≈ 1.6 π [rad]

Carrier   c(t)   s(t)   Modulated signal

12

PM   Programming Example

- I = $x(t) = A_c \cos[D_p\, m(t)]$ $= \cos[D_p \sin(2\pi f_m t)]$
- Q = $y(t) = A_c \sin[D_p\, m(t)]$ $= \sin[D_p \sin(2\pi f_m t)]$
  - A_c: Carrier signal amplitude = 1
  - Dp: Peak phase deviation

```
clear all, close all, clc

fm = 1;                    % Modulating frequency [kHz]
Deviation = 5;             % Peak phase deviation [kHz]
CSVfile = 'PM1kHzDeviation5rad.csv'

DataPoints = 1000;
Fs = fm*DataPoints         % Sampling rate [kHz]

t = 0:DataPoints-1;
I = cos(Deviation*sin(2*pi*fm*t/DataPoints));
Q = sin(Deviation*sin(2*pi*fm*t/DataPoints));
data = [I' Q'];
csvwrite(CSVfile,data);
```
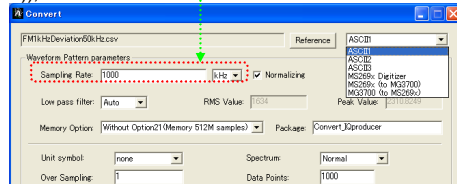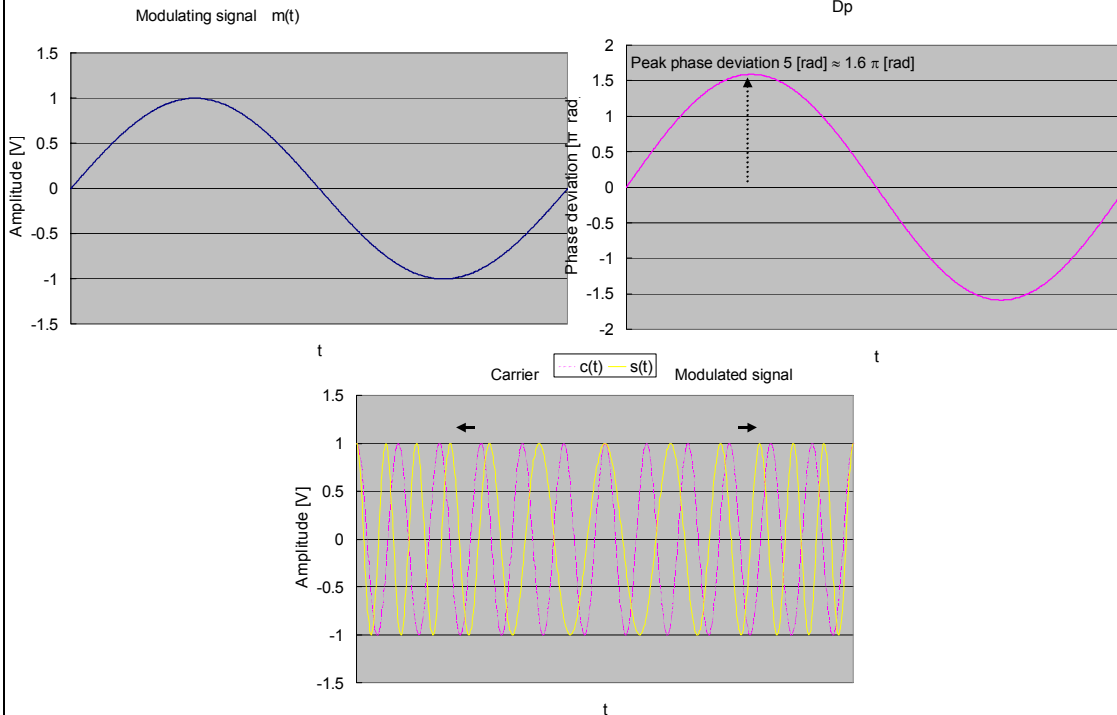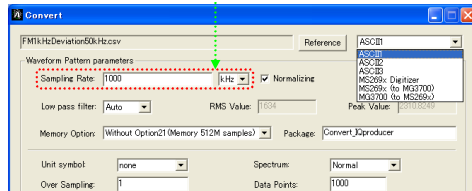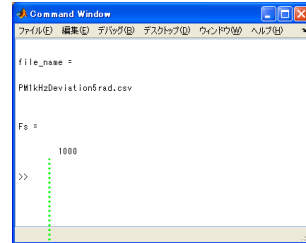

Pulse Modulation

- Pulse modulation is used for radar and satellite signals using a combination with either FM or PM, and time-multiplexed burst signals.
  - PRF/PRR: Pulse repetition frequency, rate [pps], [pulse/s]
  - Pulse period [s]: 1/PRF
  - Pulse width, duration [s]: Including rise time
  - Duty ratio, cycle [%]: Pulse width/Pulse period
  - Pulses per burst [pulses/burst]
  - Burst length [s]
  - Burst period [s]
  - Rise time [s]

13

Pulse Modulation   Programming Example

CW
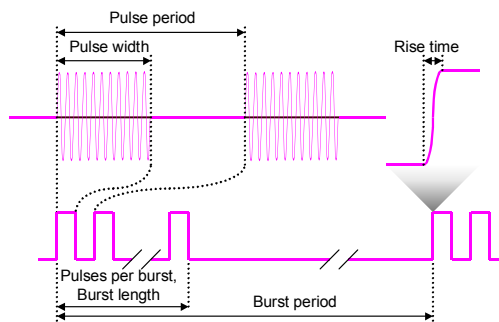
- I = $x(t) = Ac = 1$
- Q = $y(t) = Ac = 1$
  - *Ac*: Carrier signal amplitude = 1
  - PRF: 700 [pps]
  - Pulse width: 1 [μs]
  - Pulses/burst: 18
  - Burst period: 10 [s]

```
clear all, close all, clc

PRF = 700;              % [pps]
PulseWidth = 1E-6;      % [s]
PulsesPerBurst = 18;    % [pulses/burst]
BurstPeriod = 10;       % [s]
CSVfile = 'Pulse700ppsW1us18ppbB10s.csv'
zerofile = 'Pulse700ppsW1us18ppbB10sZero.csv'

Fs = LCM(PRF,1/PulseWidth)      % Sampling rate [Hz], LCM: Least Common Multiple
% Number of samples
W = PulseWidth*Fs;
PulseGap = (1/PRF-PulseWidth)*Fs;

(Continued on the next page)
```

MG3700A-E-F-11

Anritsu

Pulse Modulation   Programming Example

(Continued from previous page)

RF gate flag

I/Q
(DC)
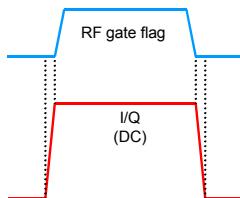
```
I = [ ones(1,W+2) zeros(1,PulseGap-2) ];
Q = [ ones(1,W+2) zeros(1,PulseGap-2) ];
RFgateFlag = [ zeros(1,1) ones(1,W) zeros(1,PulseGap) ];
Burst = [ zeros(1,1) ones(1,W) ];
for n = 2:PulsesPerBurst
    I = [ I ones(1,W+2) zeros(1,PulseGap-2) ];
    Q = [ Q ones(1,W+2) zeros(1,PulseGap-2) ];
    Burst = [ Burst ones(1,PulseGap+W) ];
    if n == PulsesPerBurst
            RFgateFlag = [ RFgateFlag ones(1,W) zeros(1,PulseGap-1) ];
    else
            RFgateFlag = [ RFgateFlag ones(1,W) zeros(1,PulseGap) ];
    end
end

Marker1 = [ Burst zeros(1,PulseGap-1) ];        % Burst
Marker2 = RFgateFlag;                            % Pulse
Marker3 = RFgateFlag;                            % Pulse
data = [I' Q' Marker1' Marker2' Marker3' RFgateFlag'];
csvwrite(CSVfile,data);

BurstGapPulseLength = PRF*BurstPeriod-PulsesPerBurst;
zero = zeros(1,Fs/PRF);  % samples/pulse
zerodata = [zero' zero' zero' zero' zero' zero'];
csvwrite(zerofile,zerodata);
zerofileMultiple = PRF*BurstPeriod-PulsesPerBurst  % Pulse length in Burst gap
```

MG3700A-E-F-11

Anritsu

Pulse Chirp Modulation   Programming Example

- I = $x(t) = A_c \cos\left[D_f \int_{-\infty}^{t} m(\sigma)d\sigma\right]$
- Q = $y(t) = A_c \sin\left[D_f \int_{-\infty}^{t} m(\sigma)d\sigma\right]$

FM

- Ac: Carrier signal amplitude = 1
- PRF: 2000 [pps]
- Pulse width: 20 [μs]
- Pulses/burst: 20
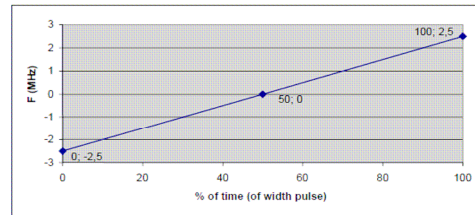
```
clear all, close all, clc

PW = 20;              % Pulse Width [us]
PRF = 2000;           % [pps]
Fpeak = 2.5;          % Peak Frequency Deviation [MHz]
CSVfile = 'RTS6W20us2000pps.csv'
zerofile ='RTS6W20us2000ppsZero.csv'

Fs = Fpeak * 40       % Sampling rate [MHz]
DataPoints = PW * Fs;
```

(Continued on the next page)

Pulse Chirp Modulation   Programming Example

(Continued from previous page)

RF gate flag

I/Q
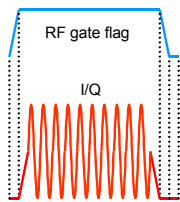
```
t = 0:DataPoints;
dPhi = 2*pi*(-Fpeak + 2* Fpeak * t/DataPoints)/Fs;   % Frequency deviation from -Fpeak to
    +Fpeak
Phi = cumsum(dPhi,2);     % Comulative Sum
I = cos(Phi);
Q = sin(Phi);

I = [zeros(1,1) I zeros(1,2)];
Q = [zeros(1,1) Q zeros(1,2)];
BurstFlag = ones(1,DataPoints+4);
RFgateFlag = [ones(1,DataPoints+3) zeros(1,1)];

data=[I' Q' BurstFlag' RFgateFlag' RFgateFlag' RFgateFlag'];     % Marker1: Burst On/Off
    Flag, Marker2&3: RF On/Off Flag
csvwrite(CSVfile,data);

zero = zeros(1,DataPoints+4);
zerodata = [zero' zero' zero' zero' zero' zero'];
csvwrite(zerofile,zerodata);

PulsePeriod = Fs * 1E6 / PRF;   % Pulse period samples
FrameLength = DataPoints + 4
GapLength = PulsePeriod - DataPoints - 4
```

16

Pulse Chirp Modulation　Programming Example

Sequence feature technique

Pulse Chirp Modulation　Programming Example

Create a sequence file

PSK, QAM

- Digital modulation schemes change the amplitude, frequency and phase of the carrier at regular time intervals, and sends a digital baseband signal.
  - » Analog modulation schemes change the carrier amplitude, frequency and phase continuously.

  Basic digital modulation schemes:
  - » Amplitude Shift Keying (ASK)
    - – The amplitude of the carrier varies linearly with the symbol sequence.
  - » Frequency Shift Keying (FSK)
    - – The frequency of the carrier varies linearly with the symbol sequence.
  - » Phase Shift Keying (PSK)
    Differential Phase Shift Keying (DPSK)
    - – The phase of the carrier varies linearly with the symbol sequence.
  - » Quadrature Amplitude Modulation (QAM)
    - – The amplitude and phase of the carrier varies linearly with the symbol sequence.
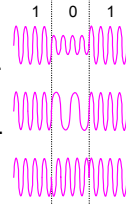
PSK, QAM

- This chapter shows common examples for QPSK, π/4DQPSK, 8PSK, 16QAM.
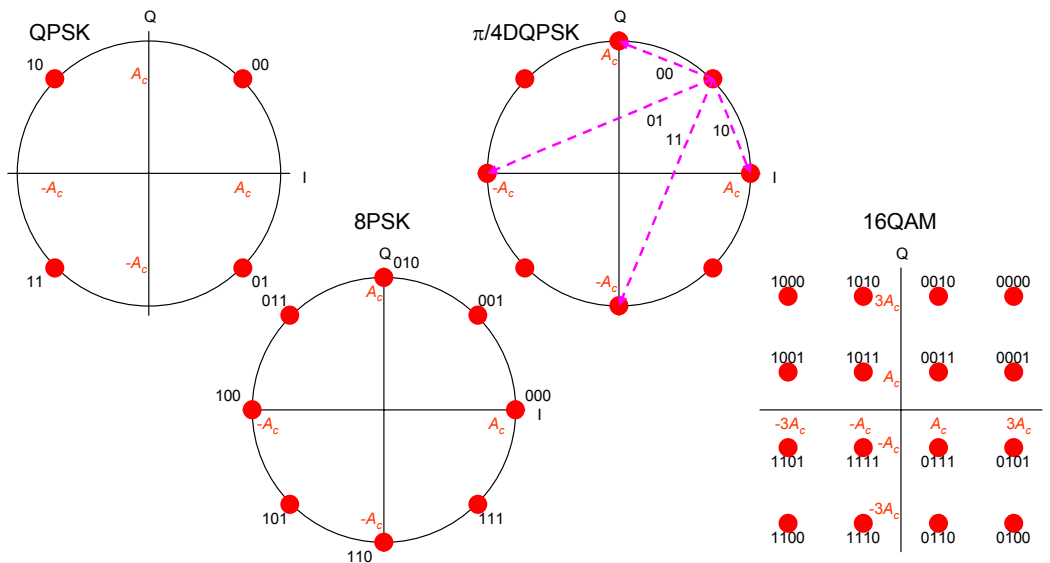
18

QPSK

Complex envelop
- $g(t) = Ac[m_1(t)+jm_2(t)] = I(t)+jQ(t)$
  - Ac: Carrier signal amplitude = 1

MATLAB requires:
Signal Processing Toolbox
Communications Toolbox

```
clear all, close all, clc

Fd = 1;        % Symbol rate [sps]
Fs = 10*Fd;    % Oversampling rate
Pd = 256;      % Symbol points
repetition = 2;              % Repetition factor of data
R = 0.5;       % Rolloff factor for FIR filter
delay = 3;     % Filter's group delay
CSVfile='QPSK.csv'

M = 4;         % Point signal constellations
SymbolData = randint(Pd,1,M);        % Random data [Communications Toolbox]
% Duplicate the symbol data because of waveform phase continuity
DuplicateSymbolData = repmat(SymbolData,repetition,1);        % Replicate the symbol
    data [Fixed-Point Toolbox]
g = pskmod(DuplicateSymbolData,M,pi/4);        % Complex envelop g(t), Initial phase
    pi/4 rad [Communications Toolbox]
```

(Continued on the next page)

/Inritsu

QPSK

(Continued from previous page)


```
filtering = rcosflt(g,Fd,Fs,'fir/normal',R,delay);        % Filter input signal using FIR filter
    [Communications Toolbox]
OversampledData = filtering(Fs*delay*2+1:Fs*delay*2+Fs*Pd);        % Correct filter delay
data = [real(OversampledData) imag(OversampledData)];        % I/Q data
csvwrite(CSVfile,data);

scatterplot(OversampledData,1,0,'b-'),hold on,plot(g,'r*'),grid        % Signal constellation

figure        % Complex envelop
t_g = 0:Pd*repetition-1;    % g(t) symbol number
t_o = delay:Fd/Fs:delay+Pd-1/Fs;        % Oversampled symbol number
subplot(2,1,1),stem(t_g,real(g),'r*'),hold on,plot(t_o,real(OversampledData)),grid% In-phase
axis([-1 Pd*repetition+1 round(min(real(g))) round(max(real(g)))])
title('I signal'),xlabel('Symbol'),ylabel('Amplitude')
subplot(2,1,2),stem(t_g,imag(g),'r*'),hold on,plot(t_o,imag(OversampledData)),grid
        % Quadrature-phase
axis([-1 Pd*repetition+1 round(min(imag(g))) round(max(imag(g)))])
title('Q signal'),xlabel('Symbol'),ylabel('Amplitude')
```

/Inritsu

19

Waveform Phase Continuity

- The MG3700A plays back a waveform of finite in length and repeats it continuously. Phase discontinuity between the end of one waveform and the start of the next repetition can lead to periodic spectral regrowth and distortion.
- Repetitions with abrupt phase changes result in high frequency spectral regrowth.

For Pd = 16 symbol points/waveform

π/4DQPSK

Complex envelop
- $g(t) = Ac[m_1(t)+jm_2(t)] = I(t)+jQ(t)$
  - *Ac*: Carrier signal amplitude = 1

MATLAB requires:
Signal Processing Toolbox
Communications Toolbox

```
clear all, close all, clc

Fd = 1;        % Symbol rate [sps]
Fs = 10*Fd;    % Oversampling rate
Pd = 256;      % Symbol points
repetition = 2;              % Repetition factor of data
R = 0.5;       % Rolloff factor for FIR filter
delay = 3;     % Filter's group delay
CSVfile='pi4DQPSK.csv'

M = 4;         % Point signal constellations
SymbolData = randint(Pd,1,M);        % Random data [Communications Toolbox]
% Duplicate the symbol data because of waveform phase continuity
DuplicateSymbolData = repmat(SymbolData,repetition,1);        % Replicate the symbol
    data [Fixed-Point Toolbox]
g = dpskmod(DuplicateSymbolData,M,pi/4);        % Complex envelop g(t) using phase
    shift pi/4 rad [Communications Toolbox]
```

(Continued on the next page)

20

π/4DQPSK

(Continued from previous page)

```
filtering = rcosflt(g,Fd,Fs,'fir/normal',R,delay);          % Filter input signal using FIR filter
    [Communications Toolbox]
OversampledData = filtering(Fs*delay*2+1:Fs*delay*2+Fs*Pd);     % Correct filter delay
data = [real(OversampledData) imag(OversampledData)];          % I/Q data
csvwrite(CSVfile,data);

scatterplot(OversampledData,1,0,'b-'),hold on,plot(g,'r*'),grid     % Signal constellation

figure          % Complex envelop
t_g = 0:Pd*repetition-1;    % g(t) symbol number
t_o = delay:Fd/Fs:delay+Pd-1/Fs;          % Oversampled symbol number
subplot(2,1,1),stem(t_g,real(g),'r*'),hold on,plot(t_o,real(OversampledData)),grid% In-phase
axis([-1 Pd*repetition+1 min(real(OversampledData)) max(real(OversampledData))])
title('I signal'),xlabel('Symbol'),ylabel('Amplitude')
subplot(2,1,2),stem(t_g,imag(g),'r*'),hold on,plot(t_o,imag(OversampledData)),grid
          % Quadrature-phase
axis([-1 Pd*repetition+1 min(imag(OversampledData)) max(imag(OversampledData))])
title('Q signal'),xlabel('Symbol'),ylabel('Amplitude')
```

8PSK

Complex envelop
- $g(t) = Ac[m_1(t)+jm_2(t)] = I(t)+jQ(t)$
  - Ac: Carrier signal amplitude = 1

MATLAB requires:
◆ Signal Processing Toolbox
◆ Communications Toolbox
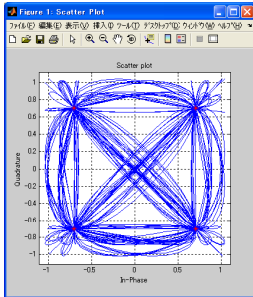
```
clear all, close all, clc

Fd = 1;       % Symbol rate [sps]
Fs = 10*Fd;   % Oversampling rate
Pd = 256;     % Symbol points
repetition = 2;             % Repetition factor of data
R = 0.5;      % Rolloff factor for FIR filter
delay = 3;    % Filter's group delay
CSVfile='8PSK.csv'

M = 8;        % Point signal constellations
SymbolData = randint(Pd,1,M);         % Random data [Communications Toolbox]
% Duplicate the symbol data because of waveform phase continuity
DuplicateSymbolData = repmat(SymbolData,repetition,1);         % Replicate the symbol
    data [Fixed-Point Toolbox]
g = pskmod(DuplicateSymbolData,M); % Complex envelop g(t) [Communications Toolbox]
```

(Continued on the next page)

8PSK

```
filtering = rcosflt(g,Fd,Fs,'fir/normal',R,delay);        % Filter input signal using FIR filter
    [Communications Toolbox]
OversampledData = filtering(Fs*delay*2+1:Fs*delay*2+Fs*Pd);      % Correct filter delay
data = [real(OversampledData) imag(OversampledData)];       % I/Q data
csvwrite(CSVfile,data);

scatterplot(OversampledData,1,0,'b-'),hold on,plot(g,'r*'),grid      % Signal constellation
figure        % Complex envelop
t_g = 0:Pd*repetition-1;     % g(t) symbol number
t_o = delay:Fd/Fs:delay+Pd-1/Fs;        % Oversampled symbol number
subplot(2,1,1),stem(t_g,real(g),'r*'),hold on,plot(t_o,real(OversampledData)),grid% In-phase
axis([-1 Pd*repetition+1 min(real(OversampledData)) max(real(OversampledData))])
title('I signal'),xlabel('Symbol'),ylabel('Amplitude')
subplot(2,1,2),stem(t_g,imag(g),'r*'),hold on,plot(t_o,imag(OversampledData)),grid
        % Quadrature-phase
axis([-1 Pd*repetition+1 min(imag(OversampledData)) max(imag(OversampledData))])
title('Q signal'),xlabel('Symbol'),ylabel('Amplitude')
```
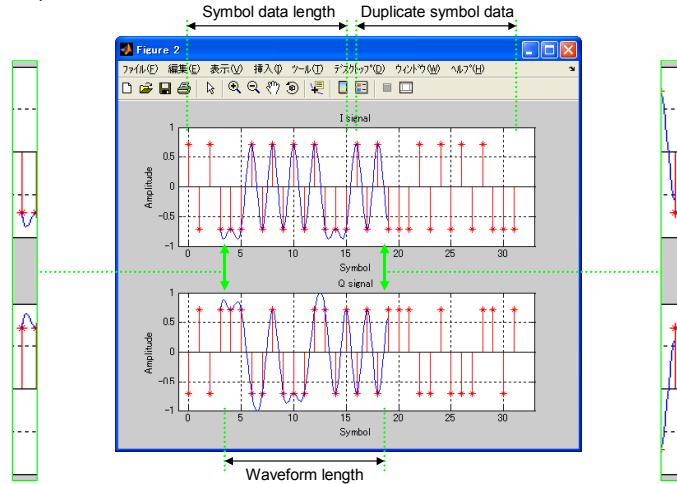
Anritsu

16QAM

Complex envelop
- $g(t) = Ac[m_1(t)+jm_2(t)] = I(t)+jQ(t)$
  - Ac: Carrier signal amplitude = 1

MATLAB requires:
Signal Processing Toolbox
Communications Toolbox

```
clear all, close all, clc

Fd = 1;        % Symbol rate [sps]
Fs = 10*Fd;% Oversampling rate
Pd = 256;    % Symbol points
repetition = 2;                % Repetition factor of data
R = 0.5;       % Rolloff factor for FIR filter
delay = 3;    % Filter's group delay
CSVfile='16QAM.csv'

M = 16;        % Point signal constellations
SymbolData = randint(Pd,1,M);        % Random data [Communications Toolbox]
% Duplicate the symbol data because of waveform phase continuity
DuplicateSymbolData = repmat(SymbolData,repetition,1);                % Replicate the symbol
    data [Fixed-Point Toolbox]
g = pskmod(DuplicateSymbolData,M); % Complex envelop g(t) [Communications Toolbox]
```
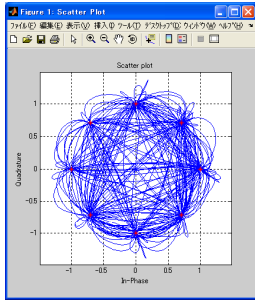
Anritsu

16QAM

(Continued from previous page)

```
filtering = rcosflt(g,Fd,Fs,'fir/normal',R,delay);        % Filter input signal using FIR filter
    [Communications Toolbox]
OversampledData = filtering(Fs*delay*2+1:Fs*delay*2+Fs*Pd);    % Correct filter delay
data = [real(OversampledData) imag(OversampledData)];     % I/Q data
csvwrite(CSVfile,data);

scatterplot(OversampledData,1,0,'b-'),hold on,plot(g,'r*'),grid    % Signal constellation
figure        % Complex envelop
t_g = 0:Pd*repetition-1;    % g(t) symbol number
t_o = delay:Fd/Fs:delay+Pd-1/Fs;        % Oversampled symbol number
subplot(2,1,1),stem(t_g,real(g),'r*'),hold on,plot(t_o,real(OversampledData)),grid% In-phase
axis([-1 Pd*repetition+1 min(real(OversampledData)) max(real(OversampledData))])
title('I signal'),xlabel('Symbol'),ylabel('Amplitude')
subplot(2,1,2),stem(t_g,imag(g),'r*'),hold on,plot(t_o,imag(OversampledData)),grid
            % Quadrature-phase
axis([-1 Pd*repetition+1 min(imag(OversampledData)) max(imag(OversampledData))])
title('Q signal'),xlabel('Symbol'),ylabel('Amplitude')
```

Simulink
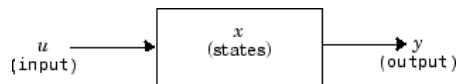
- **What is Simulink?**
  - » Simulink® is a software package for modeling, simulating, and analyzing dynamic systems. It supports linear and nonlinear systems, modeled in continuous time, sampled time, or a hybrid of the two.
- **Tool for simulation**
  - » You can easily build models from scratch, or take an existing model and add to it. It offers instant access to all the analysis tools in MATLAB, so the results can be taken and analyzed and visualized.
- **Tool for model-based design**
  - » For modeling, Simulink offers a GUI for building models as block diagrams, using click-and-drag mouse operations. You can also customize and create your own blocks.
  - » Models are hierarchical, so they can be built using both top-down and bottom-up approaches.
  - » After defining a model, you can simulate it. Simulation results can be put in the <u>MATLAB workspace</u> for post-processing and visualization.
        CSV file

For more information on Simulink, visit the MathWorks website.
http://www.mathworks.com/access/helpdesk/help/toolbox/simulink/

Simulink Demos

- Simulink demos illustrate useful modeling concepts. Access demos from the MATLAB Command Window.
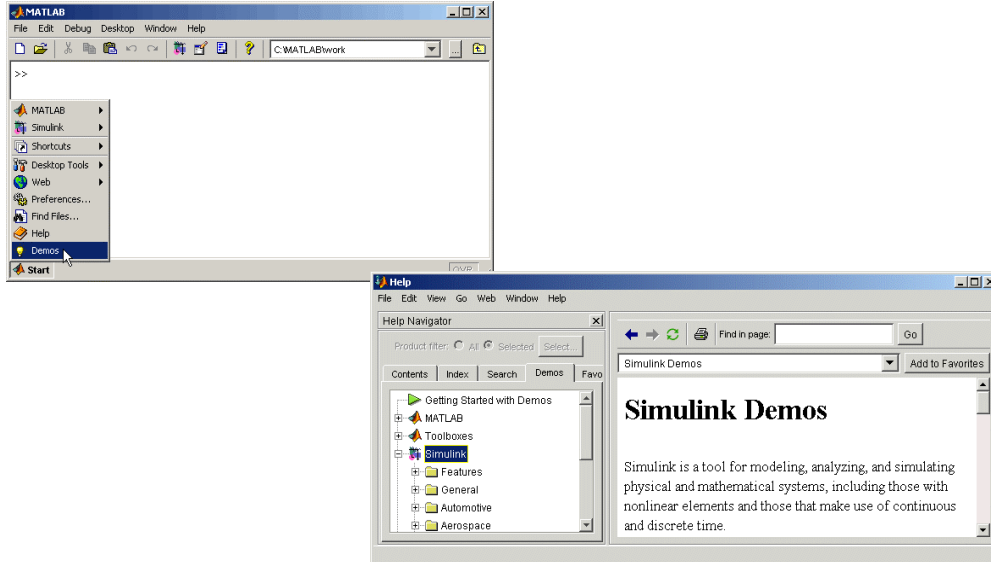
Simulink Blockset Demos

- The Blocksets are the collections of specialized Simulink blocks designed for design and simulation in a particular field.
- For example, the Communications Blockset extends Simulink with a comprehensive library of blocks to design and simulate the physical layer of communication systems and components. The blockset helps design for communications systems and their semiconductor components, such as commercial or defense wireless and wire systems.
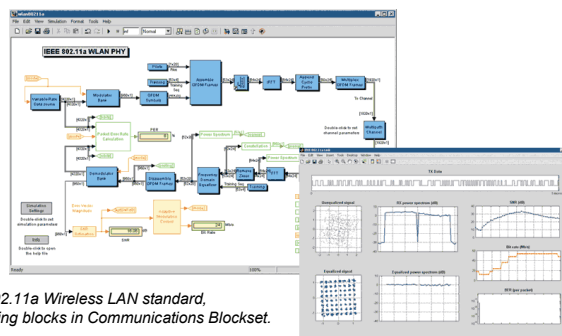
Model of physical layer of IEEE 802.11a Wireless LAN standard, including adaptive modulation, using blocks in Communications Blockset.

Simulation Data Save Technique

- Simulink Library Browser
  - » The *Math Operations* library contains blocks for modeling general mathematical functions.
  - » The *Sinks* library contains blocks that display or write block output.
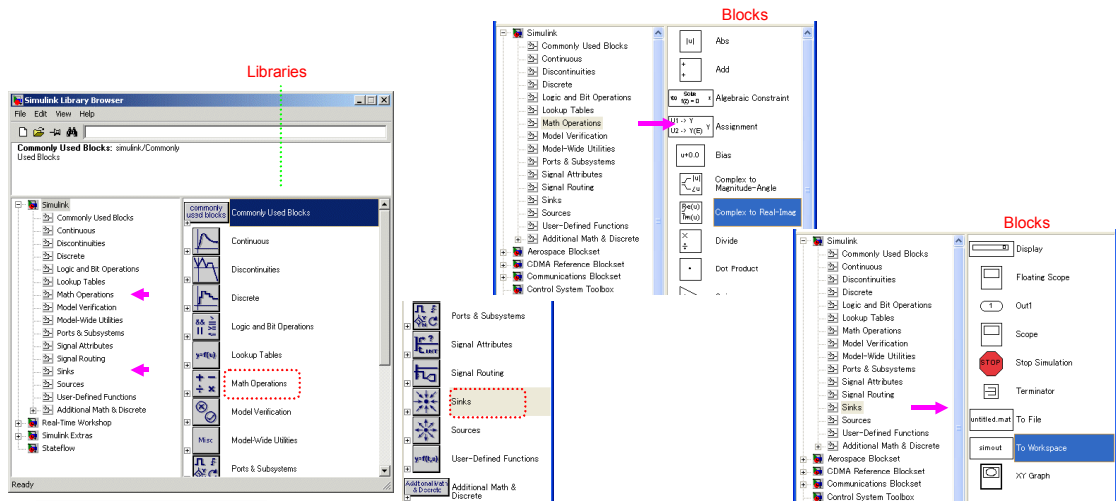
Complex to Real-Imag

- Output real and imaginary parts of complex input signal
  - – The Complex to Real-Imag block accepts a complex-valued signal of any data type supported by Simulink, including fixed-point data types. It outputs the real and/or imaginary part of the input signal, depending on the setting of the **Output** parameter. The real outputs are of the same data type as the complex input. The input can be an array (vector or matrix) of complex signals, in which case the output signals are arrays of the same dimensions. The real array contains the real parts of the corresponding complex input elements. The imaginary output similarly contains the imaginary parts of the input elements.
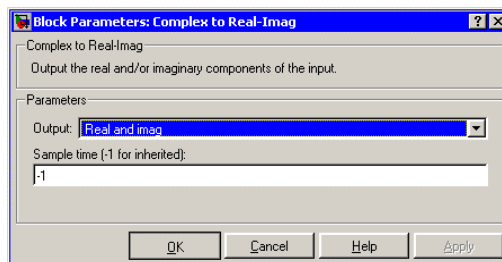
Complex to Real-Imag

- Parameters
    - » Output
        - This parameter determines the output of this block. Choose from the following values: **Real and imag** (outputs the input signal's real and imaginary parts), Real (outputs the input's real part), Imag (outputs the input's imaginary part).
    - » Sample time (-1 for inherited)
        - This parameter specifies the time interval between samples. To inherit the sample time, set this parameter to **-1**.
        - -1
        - If the block is not in a triggered subsystem, this setting specifies that the block inherits its sample time from the block connected to its input (inheritance) or, in some cases, from the block connected to its output (back inheritance). If the block is in a triggered subsystem, set the SampleTime parameter to this setting.
        Note that specifying sample-time inheritance for a source block can cause Simulink to assign an inappropriate sample time to the block if the source drives more than one block. For this reason, avoid specifying sample-time inheritance for source blocks. If it is specified, Simulink displays a warning message when updating or simulating the model.

To Workspace

- Write data to workspace
    - The To Workspace block writes its input to the workspace. The block writes its output to an array or structure with the name specified by the block's **Variable name** parameter. The **Save format** parameter determines the output format.
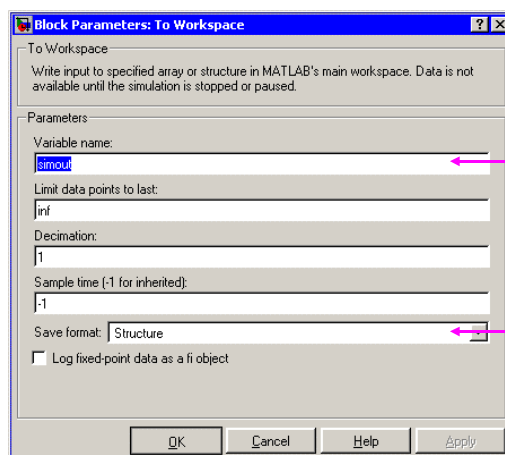
*inf: IEEE arithmetic representation for positive infinity

26

To Workspace

» **Array**

– Selecting this option causes the To Workspace block to save the input as an N-dimensional array where N is one more than the number of dimensions of the input signal. For example, if the input signal is a 1-D array (i.e., a vector), the resulting workspace array is two-dimensional. If the input signal is a 2-D array (i.e., a matrix), the array is three-dimensional.

– The way samples are stored in the array depends on whether the input signal is a scalar or vector or a matrix. If the input is a scalar or a vector, each input sample is output as a row of the array. For example, suppose that the name of the output array is simout. Then, simout(1,:) corresponds to the first sample, simout(2,:) corresponds to the second sample, etc. If the input signal is a matrix, the third dimension of the workspace array corresponds to the values of the input signal at the specified sampling point. For example, suppose again that simout is the name of the resulting workspace array. Then, simout(:,:,1) is the value of the input signal at the first sample point; simout(:,:,2) is the value of the input signal at the second sample point; etc.

– Block parameters control when and how much data the To Workspace block writes:

1. The **Limit data points to last** parameter indicates how many sample points to save. If the simulation generates more data points than the specified maximum, the simulation saves only the most recently generated samples. To capture all the data, set this value to *inf*.

2. The **Decimation** parameter allows you to write data at every nth sample, where n is the decimation factor. The default decimation, 1, writes data at every time step.

3. The **Sample time** parameter allows you to specify a sampling interval at which to collect points. This parameter is useful when you are using a variable-step solver where the interval between time steps might not be the same. The default value of -1 causes the block to inherit the sample time from the driving block when determining the points to write.

To Workspace

– For variable-step solvers, the **Output options** found on the **Data Import/Export** pane of the Configuration Parameters dialog box determine the original amount of data available to the To Workspace block. For example, to ensure that data is written at identical time points over multiple simulations, select the Produce specified output only option in the Configuration Parameters dialog box and enter the desired time vector. The To Workspace block begins with this specified time vector and further limits the amount of data written to the workspace based on its block parameters.

– During the simulation, the block writes data to an internal buffer. When the simulation is completed or paused, that data is written to the workspace. Its icon shows the name of the array to which the data is written.

To Workspace

- Parameters
  - » Variable name
    - – Name of array holding data
  - » Limit data points to last
    - – Maximum number of input samples saved (The default is *inf* samples.)
  - » Decimation
    - – Decimation factor (The default is 1.)
  - » Sample time
    - – Sample time at which to collect points
  - » Save format
    - – Format in which to save simulation output to workspace (The default is *structure*.)
  - » Log fixed-point data as a *fi* object
    - – Select to log fixed-point data to MATLAB workspace as Simulink Fixed-Point *fi* object. Otherwise, fixed-point data is logged to the workspace as double.

Simulation Data Save Example 1

- Open completed model
  - • Communications Blockset Help
  - > Examples in Documentation
  - > Digital Modulation
  - > DQPSK Signal Constellation Points and Transitions
    - – open('/MATLAB701/help/toolbox/commblks/commblks_examples/doc_dqpsk_plot.mdl')
- The model plots the output of the DQPSK Modulator Baseband block. The image shows the possible transitions from each symbol in the DQPSK signal constellation to the next symbol.
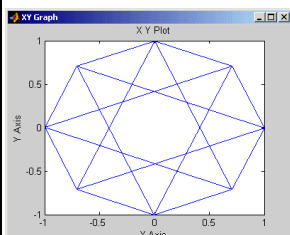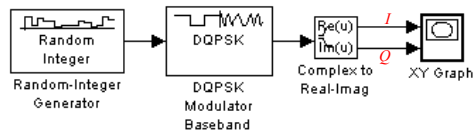
  - – Running the model produces the plot. The plot reflects the transitions among the eight DQPSK constellation points.
  - – This plot illustrates $\pi/4$DQPSK modulation, because the default **Phase offset** parameter in the DQPSK Modulator Baseband block is *pi/4*.

28

Communications Blockset Example Editing

Note

- When the MG3700A generates this I/Q signal, a raised cosine FIR filter block must be added to the model.
    - For more information about the raised cosine FIR filter block
      http://www.mathworks.com/access/helpdesk/help/toolbox/commblks/ref/raisedcosinetransmit filter.html
      – Upsample and filter input signal using raised cosine FIR filter

29

Simulation Data Save Example 2

- Open completed model
  - Communications Blockset Demos
  - > Channel Models and Impairments
  - > Rayleigh Fading Channel
  - For more information about channels
    - http://www.mathworks.com/access/helpdesk/help/toolbox/commblks/ug/fp62122.html
  - For more information about fading channels in general
    - http://www.mathworks.com/access/helpdesk/help/toolbox/comm/ug/a1069449399.html
- The model illustrates the channel's effect on a QPSK modulated signal.
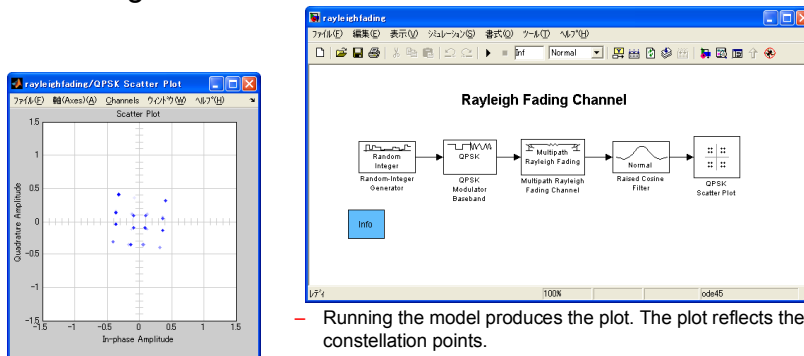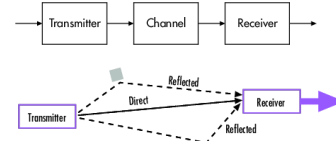
- Running the model produces the plot. The plot reflects the channel's effect on the QPSK constellation points.

Communications Blockset Demo Editing

Drag and drop

Drag and drop

30

Communications Blockset Demo Editing

Simulation Data Save Example 3

- Open the completed model
  - Communications Blockset Demos
  - \> Application-Specific Examples
  - \> IEEE 802.11a WLAN Physical Layer
- IEEE 802.11a/g WLAN OFDM physical layer model demonstrating adaptive modulation and coding
  - Requirements: Communications Toolbox, Communications Blockset, Signal Processing Blockset, Signal Processing Toolbox
  - End-to-end 802.11a physical layer
  - All mandatory and optional data rates: 6, 9, 12, 18, 24, 36, 48, and 54 Mb/s
  - BPSK, QPSK, 16QAM, 64QAM modulations
  - Forward error correction coding (convolutional; code rates 1/2, 2/3, 3/4)
  - OFDM transmission: 52 subcarriers, 4 pilots, 64-pt FFTs, CP (Cyclic Prefix)
  - Data interleaving
  - PLCP preamble (modeled as 2x2 long training sequences)
  - Receiver equalization
  - Viterbi decoding
  - Data rates selectable on-the-fly
  - Adaptive modulation demo over dispersive multipath fading channel

Communications Blockset Demo

TX Data

Equalized power spectrum

BER (per packet)

Unequalized signal

RX power spectrum

Double-click to set channel parameters

Signal Visualization

SNR   Bit rate
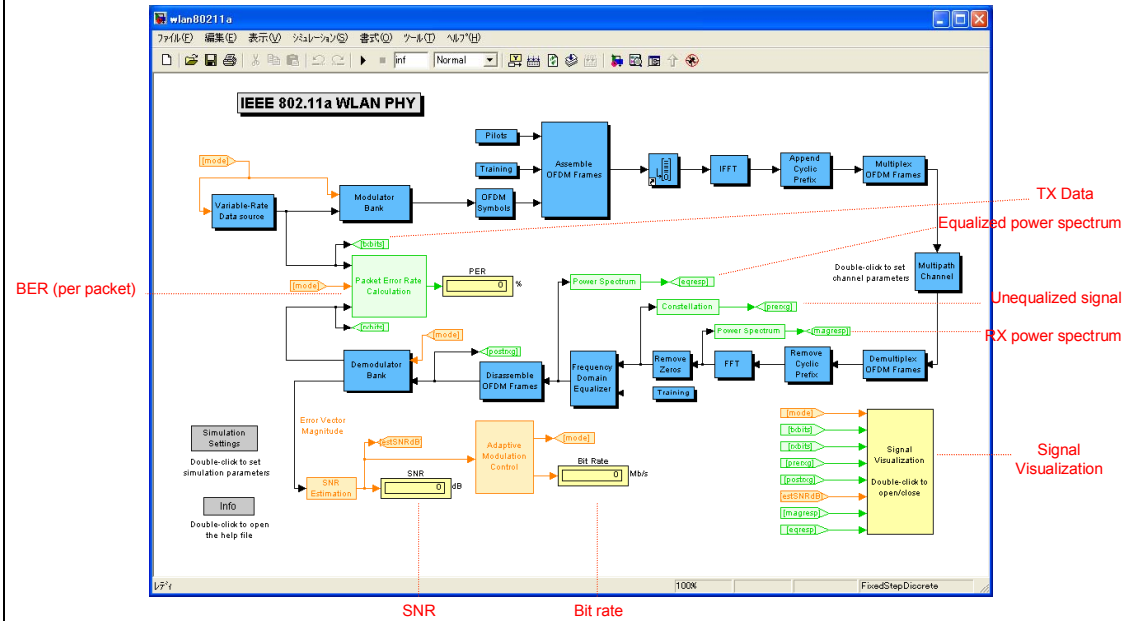
– Running the model and double-clicking Signal Visualization block produce the plot.

Communications Blockset Demo

NRZ
(Non-Return to Zero)

16QAM
in this case

PER = 1/50 (2 %)
in this case

32

Communications Blockset Demo Editing

Drag and drop

Drag and drop
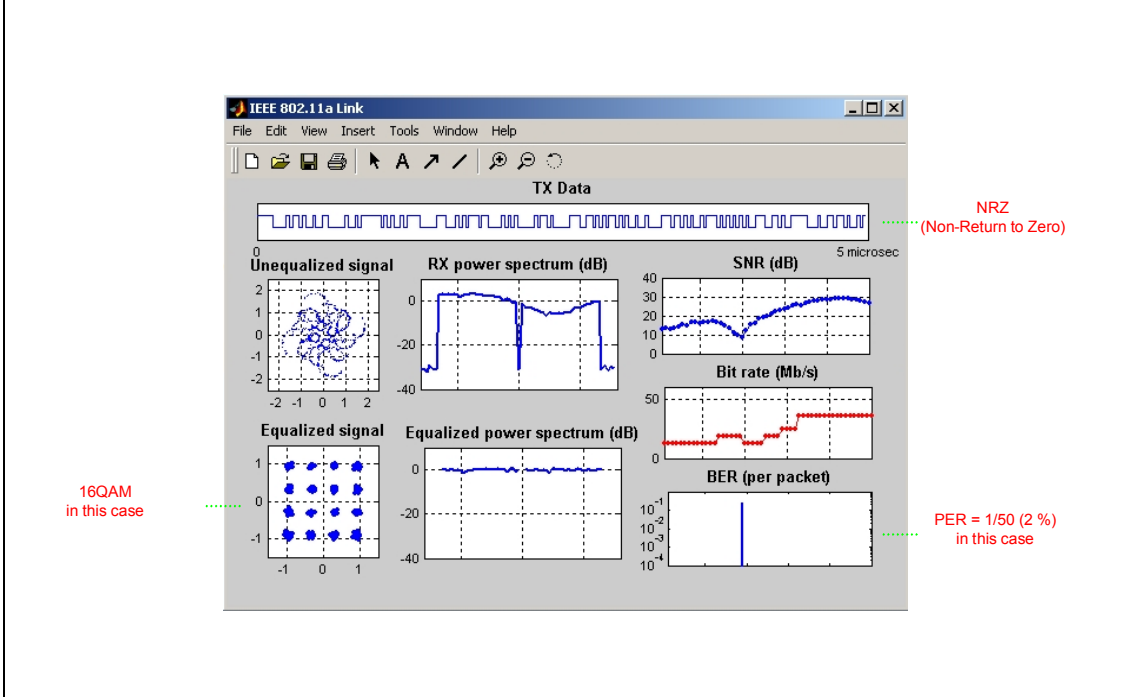
Communications Blockset Demo Editing

Start a Simulation

I or Q

Array

Measured power spectrum

Convert in IQproducer

33

Workspace Data Save

- To save the workspace variables as a CSV file, run the command by entering the following in the MATLAB command window:
  - » data = [I Q];
  - » csvwrite('IQdata.csv',data);

  - The workspace is the set of variables (named arrays) stored in memory during a MATLAB session.
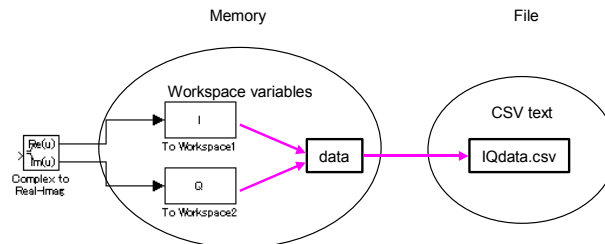
Programming Example to Convert and Transfer I/Q Data File without IQproducer

- To download IQ data to the MG3700A, typically, it is easier to use "Convert" and "Transfer & Setting" in IQproducer, because MATLAB (simulation software) and C++ (advanced programming language) can usually save the data as a CSV file.
- This process takes more time because a text file is larger, and it is a bother.
- To minimize the time to convert a IQ data file, create a specific binary data file.
- This section examines how to facilitate downloading a complex data array from within the MATLAB environment.

  1. Creating GUI with GUIDE
  2. Executing [Convert]
  3. Executing [Transfer] and [Play]

Creating GUI with GUIDE

» GUIDE, the MATLAB graphical user interface development environment, provides a set of tools for creating graphical user interfaces (GUIs). These tools simplify the process of laying out and programming GUIs.
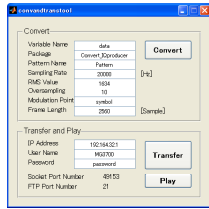
- GUI Layout
  - Using the GUIDE Layout Editor, populate a GUI by clicking and dragging GUI components, such as axes, panels, buttons, text fields, sliders, etc., into the layout area. You can also create menus and context menus for the GUI. The GUIDE saves a GUI layout to a FIG-file.
- GUI Programming
  - GUIDE automatically generates an M-file controlling how the GUI operates. This M-file provides code to initialize the GUI and contains a framework for the GUI callbacks -- the routines that execute when a user interacts with a GUI component. Using the M-file editor, add code to the callbacks to perform the required functions.

- For more information about GUIDE
  http://www.mathworks.com/access/helpdesk/help/techdoc/creating_guis/creating_guis.html

FIG-file

Executing [Convert]

Programming Flowchart

Input strings

Files

Create parameter file for converting → Parameter text file for converting

Create data file for converting → Data binary file for converting

Workspace variable

Create WVI and WVD files → File header .wvi   Data file .wvd

Delete parameter and data files

Array Editor

I   *Q*   Markers
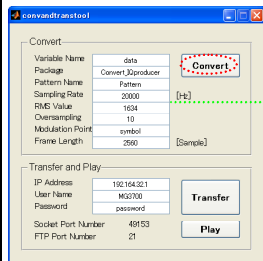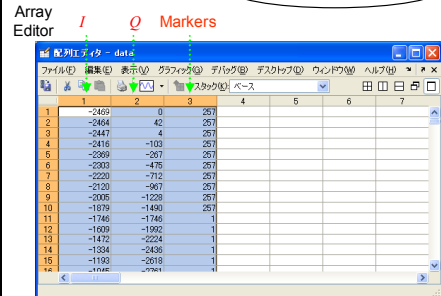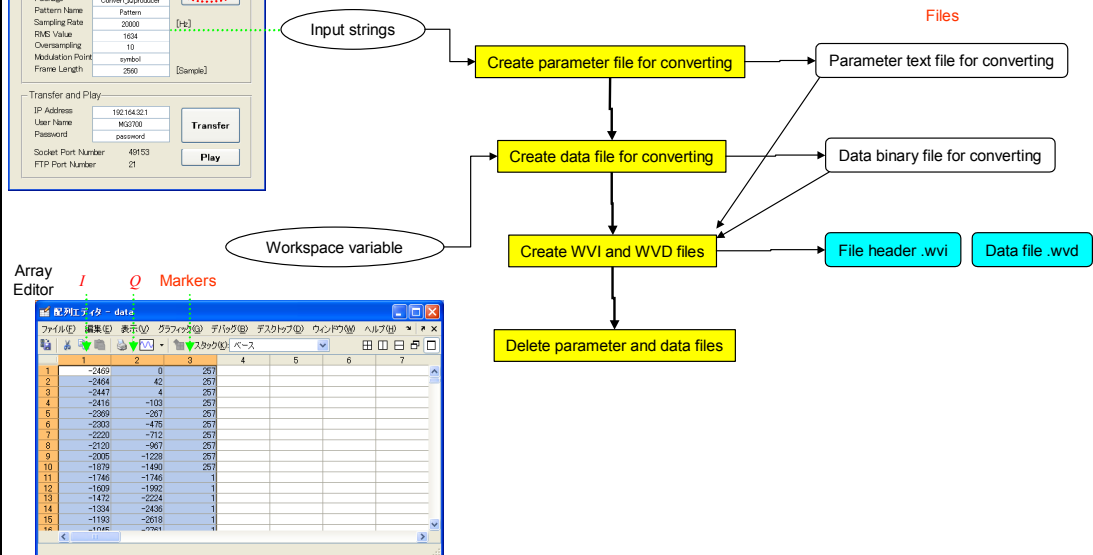
Executing [Transfer] and [Play]

Programming Flowchart

– [Transfer]: Transfer files to MG3700A HDD using FTP

Connect to MG3700A, creating FTP object

↓

Check existing package folder or make new package folder in MG3700A HDD

↓

Download WVI and WVD files
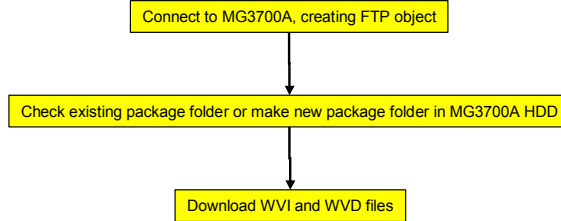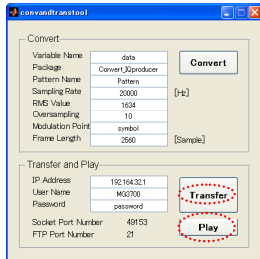
– [Play]: Load pattern data to baseband memory with TCP/IP remote control

Create TCP/IP object

↓

Load pattern data to baseband memory

↓

Play back signal pattern
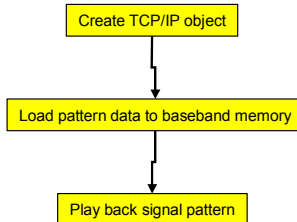
Example of Creating GUI with GUIDE

- GUIDE saves GUI layout to convandtranstool.fig.
- GUIDE automatically makes convandtranstool.m.
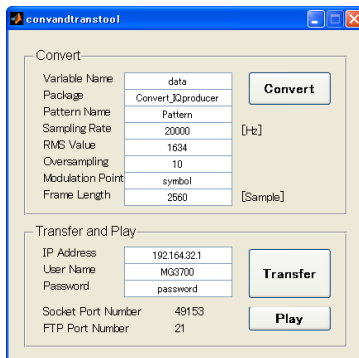
FIG-file
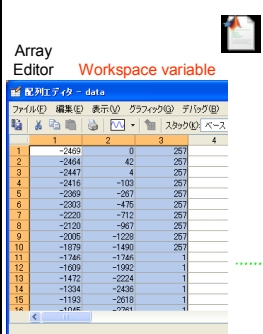
M-file function

```
function varargout = convandtranstool(varargin)
% CONVANDTRANSTOOL M-file for convandtranstool.fig
%      CONVANDTRANSTOOL, by itself, creates a new CONVANDTRANSTOOL or raises the existing
%      singleton*.
%
%      H = CONVANDTRANSTOOL returns the handle to a new CONVANDTRANSTOOL or the handle to
%      the existing singleton*.
%
%      CONVANDTRANSTOOL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in CONVANDTRANSTOOL.M with the given input arguments.
%
%      CONVANDTRANSTOOL('Property','Value',...) creates a new CONVANDTRANSTOOL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before convandtranstool_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to convandtranstool_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help convandtranstool
```

M-file

36

Call M-file Function

- M-files can be either *scripts* or *functions*. Scripts are simply files containing a sequence of MATLAB statements. Functions make use of their own local variables and accept input arguments.
- The name of a function, as defined in the first line of the M-file, should be the same as the name of the file without the .m extension.
- The variables within the body of the function are all local variables.
- When calling an M-file function from the command line or from within another M-file, MATLAB parses the function and stores it in memory. The parsed function remains in memory until cleared using the clear command or quitting MATLAB.

Array Editor   Workspace variable

I_Q = [real(OversampledData) imag(OversampledData)];   % I/Q data
.... Equal to variable "data" in previous section

RMSvalue = sqrt(sum(I_Q(:,1).^2 + I_Q(:,2).^2)/(2*length(I_Q)));
I_Q = round(I_Q*1634/RMSvalue);   % To integer .... Convert decimal fraction to 16-bit integer

marker1 = [ones(10,1) ; zeros(length(I_Q)-10,1)];
marker2 = zeros(length(I_Q),1);
marker3 = zeros(length(I_Q),1);
gate = ones(length(I_Q),1);   % RF gate flag
marker_gate = marker3*2^10 + marker2*2^9 + marker1*2^8 + gate*2^0;
data = [I_Q marker_gate];

convandtranstool   .... Call M-file function

Convert   Programming Example

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

- Create parameter file for converting

valname = get(handles.edit1, 'String');
package = get(handles.edit9, 'String');
patname = get(handles.edit2, 'String');
samprate = get(handles.edit10, 'String');
rmsval = get(handles.edit3, 'String');
oversamp = get(handles.edit4, 'String');   .... Get input strings
sysunit = get(handles.edit5, 'String');
framelen = get(handles.edit11, 'String');
if isempty(package)
    package = 'Convert_IQproducer';
End

(Continued on the next page)

Convert Programming Example

(Continued from previous page)

```
fid = fopen('wave_info.dat', 'w');          ............................    Open file, or create new file, for writing
fprintf(fid, '[Wave Info]\n');
fprintf(fid, 'Soft Type = MX3700\n');       % If using for MG3700
% fprintf(fid, 'Soft Type = MX2690\n');  % If using for MS2690
fprintf(fid, ['Package = ' package '\n']);
fprintf(fid, ['Pattern Name = ' patname '\n']);
fprintf(fid, ['Sampling Rate = ' samprate 'Hz\n']);
if ~isempty(oversamp)
    fprintf(fid, ['Over Sampling = ' oversamp '\n']);
end
if ~isempty(sysunit)
    fprintf(fid, ['System Unit = ' sysunit '\n']);
end
if ~isempty(framelen)
    fprintf(fid, ['Frame Length = ' framelen '\n']);
end
if ~isempty(rmsval)
    fprintf(fid, ['WVI RMS Value = ' rmsval '\n']);
    fprintf(fid, ['WVD RMS Value = ' rmsval '\n']);
end
fprintf(fid, 'Marker1 = Head\n');
fprintf(fid, 'Marker2 = \n');
fprintf(fid, 'Marker3 = \n');
```

```
[Wave Info]
Soft Type = MX3700
Package = Convert_IQproducer
Pattern Name = Pattern
Sampling Rate = 20000Hz
Over Sampling = 10
System Unit = symbol
Frame Length = 2560
WVI RMS Value - 1634
WVD RMS Value = 1634
Marker1 = Head
Marker2 =
Marker3 =
[Rf Gate]
Rf On Off Threshold = 1%
Min Rf Gate Length = 100samples
```
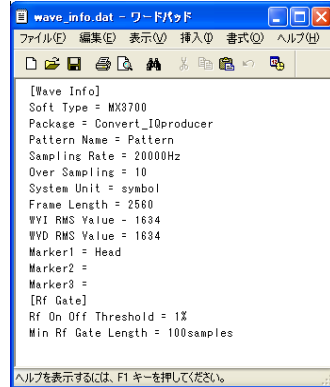
(Continued on the next page)

Convert Programming Example

(Continued from previous page)

```
% Define the following parameters in case of burst signal.
fprintf(fid, '[Rf Gate]\n');
fprintf(fid, 'Rf On Off Threshold = 1%%\n');
fprintf(fid, 'Min Rf Gate Length = 100samples\n');

fclose(fid);             ...................................................................    Close the open file
```

38

Convert　Programming Example

- Create data file for converting

  evalin('base', 'fid_r = fopen("wave_raw.dat", "w");'); ····   Open file, or create new file, for writing
  evalin('base', ['fwrite(fid_r, ' valname '", "int16");']); ····   Write binary data with integer 16 bits to file
  evalin('base', 'fclose(fid_r);'); ··   Close open file
  evalin('base', 'clear fid_r'); ····················   Remove items from workspace, freeing up system memory

- Create WVI and WVD files

  !MakeWvFile "./wave_info.dat" "./wave_raw.dat" ··········   Execute EXE file

- Delete parameter and data files

  delete('wave_info.dat');
  delete('wave_raw.dat');

Create WVI and WVD files: MakeWvFile

- The MakeWvFile.exe application software is provided by Anritsu.
- It creates the data file (.wvd) and file header (.wvi) from the specific binary data file.

- Function reference
  - » Syntax
    - – MakeWvFile "ParameterFilename" "DataFilename"
  - » Arguments
    - – "ParameterFilename"　　　　Text filename for file header (.wvi)
    - – "DataFilename"　　　　　　Specific binary data filename for data file (.wvd)

Command Prompt

Parameter text file for MakeWvFile


```
[Wave Info]
Soft Type = MX3700
Package = Convert_IQproducer
Pattern Name = Pattern
Sampling Rate = 20000Hz
Over Sampling = 10
System Unit = symbol
Frame Length = 2560
WVI RMS Value = 1634
WVD RMS Value = 1634
Marker1 = Head
Marker2 =
Marker3 =
[Rf Gate]
Rf On Off Threshold = 1%
Min Rf Gate Length = 100samples
```

- [Wave Info]
  - » Soft Type
    - MX3700
      - For MG3700A
    - MX2690
      - For MS2690A-20 (VSG option)
        - If the parameter is blank, *MX3700* is set.
  - » Package
    - Folder name for pattern file
      - ≤ 30 characters
        - If the parameter is blank, *Convert_IQproducer* is set.
  - » Pattern Name
    - ≤ 20 characters
  - » Sampling Rate
    - Number of I/Q waveform samples per second (expressed in Hz and equal to reciprocal of sampling interval)
      - 20000 to 160000000Hz (20 kHz to 160 MHz)
        - Resolution 0.001 Hz
  - » Over Sampling
    - Oversampling ratio (OSR): Ratio of sampling rate to modulation rate
      - 1 to 999
        - If the parameter is blank, *1* is set.

Discover What's Possible™
MG3700A-E-F-11

Slide 79

Anritsu
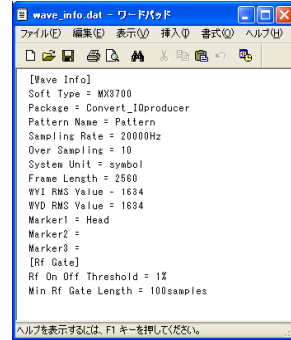
Parameter text file for MakeWvFile

- » System Unit
  - Modulation point for modulation rate i.e. chip, symbol, sample
    - ≤ 6 characters
      - If the parameter is blank, *None* is set.
- » Frame Length
  - Frame sample length
    - 1 to 8388607
      - If the parameter is blank, Input Frame Trigger cannot be used.
- » WVI RMS Value
  - I/Q DAC amplitude adjust to SG output level
    - 1 to 8191
      - *Note: In the case of Soft Type = MX2690, set *1157*.
      - If the parameter is blank, $RMS_{I/Q}$ calculated from data file is set.
- » WVD RMS Value
  - $RMS_{I/Q}$ for adjustment of I/Q amplitude in data file
    - 1 to 8191
      - If the parameter is blank, I/Q amplitude in data file is not changed.

$$* \quad RMS_{I/Q} = \sqrt{\frac{\sum_{n=1}^{N}(I_n^2 + Q_n^2)}{2N}} \qquad n: \text{Data with RF gate flag active (On)}$$

Discover What's Possible™
MG3700A-E-F-11

Slide 80

Anritsu

40

Parameter text file for MakeWvFile

> » Marker1
> » Marker2
> » Marker3
>> – Event marker name indicated on MG3700A (or MS2690A) display
>>> • ≤ 31 characters
>>>> – If the parameter is blank, it isn't indicated on MG3700A (or MS2690A) display.

Parameter text file for MakeWvFile

Using RF Gate optimizes RF gate flag to scale I/Q data within burst signal.
  In the case of no burst signal, these parameters can be skipped.
• [RF Gate]
> » Rf On Off Threshold
>> – Threshold level for automatic detection of no signal and active/inactive (On/Off) definition of RF gate flag
>>> • 0 to 100%
>>> 100% reference level: $\text{Peak}\sqrt{I_n^2 + Q_n^2}$
>>>> – If the parameter is blank, RF gate flag is not changed.
> » Min Rf Gate Length
>> – Minimum consecutive samples for automatic detection of no signal and inactive (Off) definition of RF gate flag
>>> • 0 to 100000 samples
>>>> – If the parameter is blank, RF gate flag is not changed.

RF Gate Parameters

- Blue line
  - Data (.wvd) with RF Gate
    - $RMS_{I/Q}$ (0 dB) is calculated from burst I/Q data.
- Red line
  - Data (.wvd) without RF Gate
    - $RMS_{I/Q}$ (0 dB) is calculated from all I/Q data.
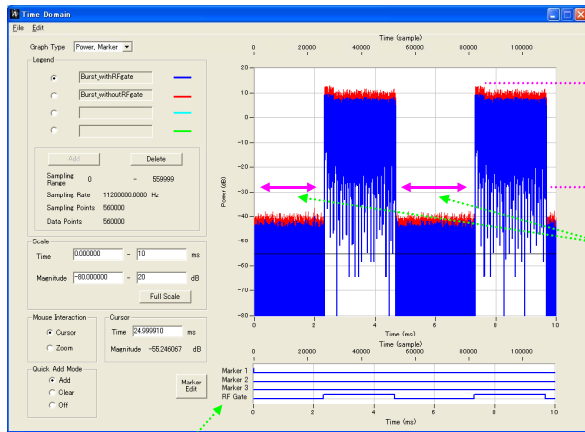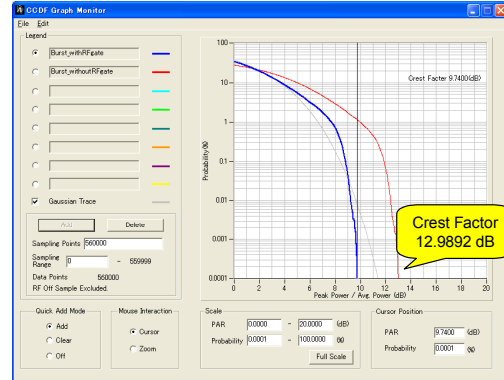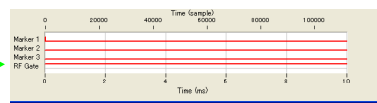
Peak level (+12.9892 dB)

Rf On Off Threshold 1% (-40 dB)

Min Rf Gate Length ≥ 100 samples

RF gate flag

Crest Factor 12.9892 dB

Data binary file for MakeWvFile

- Specific binary data file format
  - » Each data point needs 6 bytes as 2 bytes for the I point, 2 bytes for the Q point, 1 byte for three event markers and 1 byte for the RF gate flag.

6 bytes/point

Data binary file for converting → MakeWvFile → Data file .wvd

4 bytes/point

Point	I 2 bytes	Q 2 bytes	3 Event markers 1 byte	RF gate flag 1 byte	
1	16-bit signed integer	16-bit signed integer	0000 0111	0000 0001	1: Active (On)
2			0000 0000	0000 0000	0: Inactive (Off)
3					
:					

1st bit: Marker 1
2nd bit: Marker 2
3rd bit: Marker 3
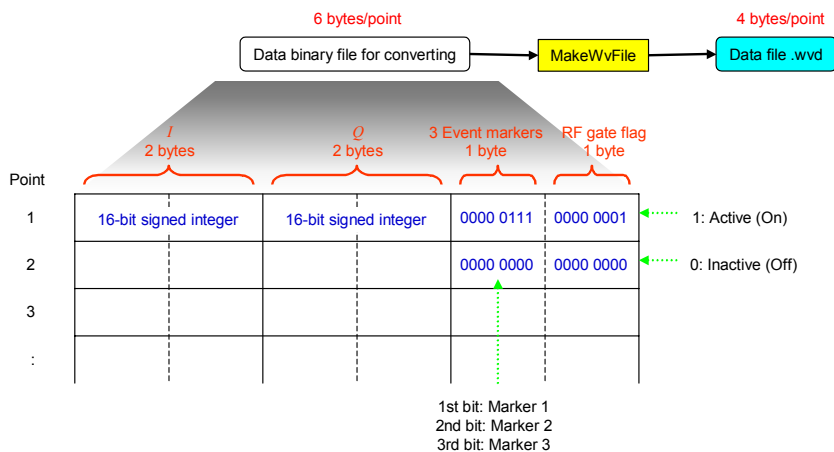
Data binary file for MakeWvFile

» I/Q binary data is 16-bit two's complement, representing signed integers.

I/Q data range	Binary data	HEX data	I/Q output voltage
– 32767	01111111 11111111	7FFF	V_{max}
⋮			
– 1	00000000 00000001	0001	
– 0	00000000 00000000	0000	0 V
– -1	11111111 11111111	FFFF	
⋮			
– -32768	10000000 00000000	8000	V_{min}

» Example for Little Endian (byte order)

Byte Swapping

5B F6

- 5B F6 00 00 01 01   60 F6 2A 00 01 01   …   [HEX]
  I   Q   Gate Markers   I   Q   Gate Markers
  -2469 0   257   -2464 42   257   …   [Decimal]

– For Big Endian (byte order) referred to as byte swapping

F6 5B

F6 5B 00 00 01 01   F6 60 00 2A 01 01   …   [HEX]
Q   I   Markers Gate   I   Q   Markers Gate

Binary Editor
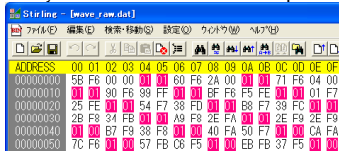
Byte Order

- The little endian or big endian byte order depends on the type of PC processor.
    » Intel and AMD processors use little endian.
    » Sun and Motorola processors use big endian.
    » The Apple PowerPC processor, while big endian oriented, also supports the little endian order.
        – Always refer to the processor manufacturer to determine the order they use for bytes and, if they support both, to understand how to ensure that you are using the correct byte order.
- The byte order describes how the system processor stores integer values as binary data in memory.
    » When outputting data from a little endian system to a text file (ASCII text), the values are the same as viewed from a big endian system.
    » The order only becomes important when using binary data, as when downloading data to MG3700A.

Byte Order

- The LSB and MSB positioning changes with byte order.
- In little endian order, the LSB and MSB are next to each other in the bit sequence.

Example of -2469 signed integer
- » Little endian
  - 5B F6 [HEX]   0 1 0 1 1 0 1 1   1 1 1 1 0 1 1 0   [Binary]

(LSB ↓ over second-to-last bit of first byte; MSB ↓ over first bit of second byte)

- » Big endian
  - F6 5B [HEX]   1 1 1 1 0 1 1 0   0 1 0 1 1 0 1 1   [Binary]

(MSB ↓ over first bit of first byte; LSB ↓ over second-to-last bit of second byte)

  - Most Significant Bit (MSB) is bit position in binary number with greatest value.
  - Least Significant Bit (LSB) is bit position in binary integer determining whether the number is even or odd.
    - In 2-byte data, the MSB appears in the second byte.

Two's Complement Integers

- Two's complement is a popular way to represent signed integers by counting backwards in computer. The MSB represents the sign of positive and negative values.

HEX	Binary	Decimal
09 A5	00001001 10100101	+2569
+		
F6 5B	11110110 01011011	-2569
	= 1 00000000 00000000	0

  - Ignoring 17th bit (leftmost bit) gives actual answer "0".

  - » The decimal value of a two's complement binary number is calculated by taking the value of the MSB, where the value is negative when the bit is one, and adding the values for each power of two where there is a one.
    - F6 5B: 11110110 01011011

$= -2^{15} + 2^{14} + 2^{13} + 2^{12} + 2^{10} + 2^9 + 2^6 + 2^4 + 2^3 + 2^1 + 2^0 = -2569$

Two's Complement Integers

- Calculating two's complement
  - » When finding the two's complement of a binary number, the bits are inverted and the value of 1 is added to the resulting value. Bit overflow is ignored.
    - Beginning with F6 5B (-2469):

      11110110 01011011
      - To convert to +2469 in two's complement notation, the bits are inverted; 0 becomes 1, and 1 becomes 0:

        00001001 10100100
        - This numeral is the one's complement of the decimal value -2469.
      - To obtain the two's complement, 1 is added:

        00001001 10100101
    - Beginning with 09 A5 (+2469):

      00001001 10100101
      - To convert to -2469 in two's complement notation, the bits are inverted; 0 becomes 1, and 1 becomes 0:

        11110110 01011010
      - To obtain the two's complement, 1 is added:

        11110110 01011011

MakeWvFile Error

- An error code is returned when an error occurs.
- Error code
  - » 0

    successful completion
  - » 16

    File writing failure
  - » 23

    Converting failure due to irregular Pattern Name in parameter text file
  - » 24

    Converting failure due to irregular Package name in parameter text file
  - » 25

    Converting failure due to irregular Sampling Rate in parameter text file
  - » 64

    WVD file opening failure
  - » 65

    WVI file opening failure
  - » 66

    Data binary file opening failure
  - » 67

    Parameter text file opening failure

Transfer　Programming Example

- Connect to MG3700A, creating FTP object

```
ipaddress = get(handles.edit6, 'String');      ...................  Get input strings
username = get(handles.edit7, 'String');
password = get(handles.edit8, 'String');

f = ftp(ipaddress, username, password);
```

Transfer　Programming Example

- Check existing package folder or make new package folder in MG3700A HDD

```
package = get(handles.edit9, 'String');      ...................  Get input strings
patname = get(handles.edit2, 'String');
if isempty(package)
    package = 'Convert_IQproducer';
end

cd(f, 'hdd0/PACKAGE');          ............................  Change directory in MG3700A HDD
buff = dir(f, package);          ............................  Directory listing
if length(buff) < 4
    mkdir(f, package);          ............................  Make new directory
end
```

- Download WVI and WVD files

```
cd(f, package);
mput(f, ['./' patname '.wvi']);          ............................  Upload files
mput(f, ['./' patname '.wvd']);

close(f);          ............................  Close FTP object
```

Play　　　Programming Example

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject   handle to pushbutton4 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles   structure with handles and user data (see GUIDATA)

- Create TCP/IP object

```
ipaddress = get(handles.edit6, 'String');          .................. Get input strings
t = tcpip(ipaddress, 49153);                        ................................. Instrument Control Toolbox
fopen(t);                                           ................................. Connect TCP/IP object to MG3700A
```

Play　　　Programming Example

- Load pattern data to baseband memory

```
package = get(handles.edit9, 'String');
patname = get(handles.edit2, 'String');            .................. Get input strings
if isempty(package)
    package = 'Convert_IQproducer';
end

fprintf(t, ['LDFILE WMA,''' package ''', ''' patname '''']);

for n = 1:150                                       ........................................ Monitor END Event Status Register
    pause(1);
    fprintf(t, 'ESR2?');
    flag = str2num(fscanf(t));
    if bitget(flag, 5) == 1
            break;
    end
end
```

- Play back signal pattern

```
fprintf(t, 'PATCOMBMODE DEFINED');                 ........................... Pattern Combination Mode: Defined
fprintf(t, ['LOADEDFILESEL WMA,''' package ''',''' patname '''']);
fprintf(t, '*OPC?');                               .................................. Operation Complete Query
fscanf(t);

fclose(t);                                         ................................................. Disconnect TCP/IP object
delete(t);
```

Available Sample Programs

Sample programs are available to self-customize them easily.

Anritsu_SG_demo.m
  – M-file to run on MATLAB

convandtranstool.m
  – M-file function called from within Anritsu_SG_demo.m

convandtranstool.fig
  – GUI layout saved by GUIDE

» MakeWvFile.exe
  – Executable file to convert a data file into Anritsu VSG format

» MG37WP06.dll
  – DLL file called from within MakeWvFile.exe

• Embedded object contains the above five files.
  » MATLAB_ProgrammedSamples2008.zzz (100 KB)

  – Change the file name into MATLAB_ProgrammedSamples2008.zip after saving the embedded file.

48

/Anritsu

Specifications are subject to change without notice.

Anritsu Corporation
5-1-1 Onna, Atsugi-shi, Kanagawa, 243-8555 Japan
Phone: +81-46-223-1111
Fax: +81-46-296-1264

• **U.S.A.**
Anritsu Company
1155 East Collins Blvd., Suite 100, Richardson,
TX 75081, U.S.A.
Toll Free: 1-800-267-4878
Phone: +1-972-644-1777
Fax: +1-972-671-1877

• **Canada**
Anritsu Electronics Ltd.
700 Silver Seven Road, Suite 120, Kanata,
Ontario K2V 1C3, Canada
Phone: +1-613-591-2003
Fax: +1-613-591-1006

• **Brazil**
Anritsu Eletrônica Ltda.
Praca Amadeu Amaral, 27 - 1 Andar
01327-010-Paraiso-São Paulo-Brazil
Phone: +55-11-3283-2511
Fax: +55-11-3288-6940

• **Mexico**
Anritsu Company, S.A. de C.V.
Av. Ejército Nacional No. 579 Piso 9, Col. Granada
11520 México, D.F., México
Phone: +52-55-1101-2370
Fax: +52-55-5254-3147

• **U.K.**
Anritsu EMEA Ltd.
200 Capability Green, Luton, Bedfordshire, LU1 3LU, U.K.
Phone: +44-1582-433200
Fax: +44-1582-731303

• **France**
Anritsu S.A.
16/18 avenue du Québec-SILIC 720
91961 COURTABOEUF CEDEX, France
Phone: +33-1-60-92-15-50
Fax: +33-1-64-46-10-65

• **Germany**
Anritsu GmbH
Nemetschek Haus, Konrad-Zuse-Platz 1
81829 München, Germany
Phone: +49-89-442308-0
Fax: +49-89-442308-55

• **Italy**
Anritsu S.p.A.
Via Elio Vittorini 129, 00144 Roma, Italy
Phone: +39-6-509-9711
Fax: +39-6-502-2425

• **Sweden**
Anritsu AB
Borgafjordsgatan 13, 164 40 KISTA, Sweden
Phone: +46-8-534-707-00
Fax: +46-8-534-707-30

• **Finland**
Anritsu AB
Teknobulevardi 3-5, FI-01530 VANTAA, Finland
Phone: +358-20-741-8100
Fax: +358-20-741-8111

• **Denmark**
Anritsu A/S
Kirkebjerg Allé 90, DK-2605 Brøndby, Denmark
Phone: +45-72112200
Fax: +45-72112210

• **Spain**
Anritsu EMEA Ltd.
Oficina de Representación en España
Edificio Veganova
Avda de la Vega, n° 1 (edf 8, pl 1, of 8)
28108 ALCOBENDAS - Madrid, Spain
Phone: +34-914905761
Fax: +34-914905762

• **United Arab Emirates**
Anritsu EMEA Ltd.
Dubai Liaison Office
P O Box 500413 - Dubai Internet City
Al Thuraya Building, Tower 1, Suit 701, 7th Floor
Dubai, United Arab Emirates
Phone: +971-4-3670352
Fax: +971-4-3688460

• **Singapore**
Anritsu Pte. Ltd.
60 Alexandra Terrace, #02-08, The Comtech (Lobby A)
Singapore 118502
Phone: +65-6282-2400
Fax: +65-6282-2533

• **India**
Anritsu Pte. Ltd.
India Branch Office
Unit No. S-3, Second Floor, Esteem Red Cross Bhavan,
No. 26, Race Course Road, Bangalore 560 001, India
Phone: +91-80-32944707
Fax: +91-80-22356648

• **P.R. China (Hong Kong)**
Anritsu Company Ltd.
Units 4 & 5, 28th Floor, Greenfield Tower, Concordia Plaza,
No. 1 Science Museum Road, Tsim Sha Tsui East,
Kowloon, Hong Kong
Phone: +852-2301-4980
Fax: +852-2301-3545

• **P.R. China (Beijing)**
Anritsu Company Ltd.
Beijing Representative Office
Room 1515, Beijing Fortune Building,
No. 5, Dong-San-Huan Bei Road,
Chao-Yang District, Beijing 10004, P.R. China
Phone: +86-10-6590-9230
Fax: +86-10-6590-9235

• **Korea**
Anritsu Corporation, Ltd.
8F Hyunjuk Building, 832-41, Yeoksam Dong,
Kangnam-ku, Seoul, 135-080, Korea
Phone: +82-2-553-6603
Fax: +82-2-553-6604

• **Australia**
Anritsu Pty. Ltd.
Unit 21/270 Ferntree Gully Road, Notting Hill,
Victoria 3168, Australia
Phone: +61-3-9558-8177
Fax: +61-3-9558-8255

• **Taiwan**
Anritsu Company Inc.
7F, No. 316, Sec. 1, Neihu Rd., Taipei 114, Taiwan
Phone: +886-2-8751-1816
Fax: +886-2-8751-1817

Please Contact:

071001

70 Printed on 70%
Recycled Paper

No. MG3700A-E-F-11-(3.00) 公知   Printed in Japan   2008-1   AKD