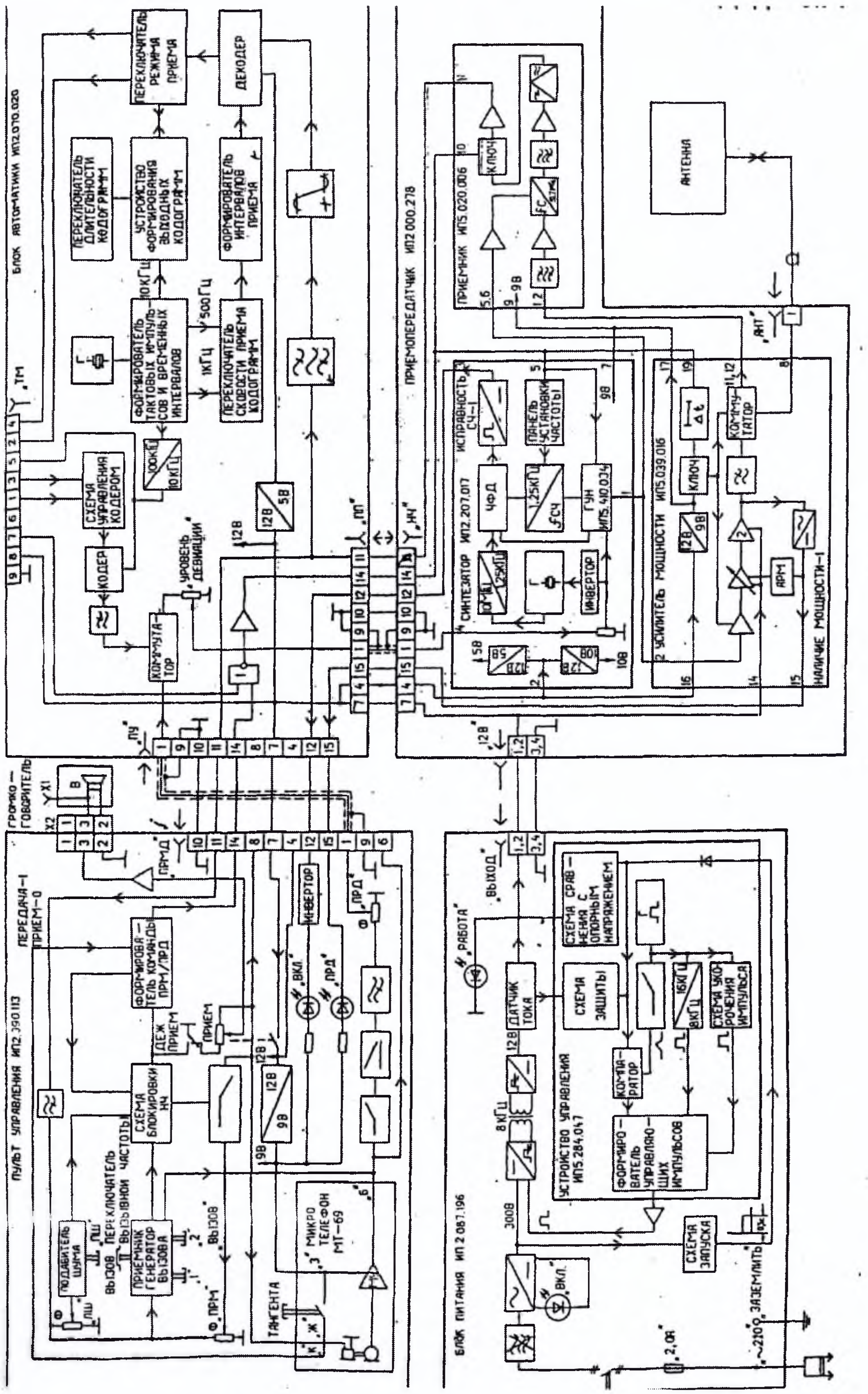
Додаток 1 Структурна схема системи «Естакада-1»


БЛОК АВТОМАТИКИ ИП2.070.020
ПЕРЕКЛЮЧАТЕЛЬ РЕЖИМА ПРИЕМА
ДЕКОДЕР
ПЕРЕКЛЮЧАТЕЛЬ ДЛИТЕЛЬНОСТИ КОДОГРАММ
УСТРОЙСТВО ФОРМИРОВАНИЯ ВЫХОДНЫХ КОДОГРАММ
ФОРМИРОВАТЕЛЬ ИНТЕРВАЛОВ ПРИЕМА
ФОРМИРОВАТЕЛЬ ТАКТОВЫХ ИМПУЛЬСОВ И ВРЕМЕННЫХ ИНТЕРВАЛОВ
10КГц
1КГц
500Гц
ПЕРЕКЛЮЧАТЕЛЬ СКВАЖНОСТИ ПРИЕМА КОДОГРАММ
"ТМ"
100КГц
10КГц
СХЕМА УПРАВЛЕНИЯ КОДЕРОМ
КОДЕР
КОММУТАТОР
УРОВЕНЬ ДЕВИАЦИИ
12В
5В
"ПЛ"
"НЧ"
"ПУ"
ПРИЕМОПЕРЕДАТЧИК ИП2.000.278
ПРИЕМНИК ИП5.020.006
КЛЮЧ
АНТЕННА
СИНТЕЗАТОР ИП2.207.017
ИСПРАВНОСТЬ СЧ-1
ПАНЕЛЬ УСТАНОВКИ ЧАСТОТЫ
ЧФД
1,25КГц
ГУН ИП5.410.034
ИНВЕРТОР
УСИЛИТЕЛЬ МОЩНОСТИ ИП5.039.016
КЛЮЧ
КОММУТАТОР
АРМ
НАЛИЧИЕ МОЩНОСТИ-1
"АНТ"
9В
ПУЛЬТ УПРАВЛЕНИЯ ИП2.390.113
ГРОМКОГОВОРИТЕЛЬ
X1
X2
В
ПЕРЕДАЧА-1 ПРИЕМ-0
"ПРМД"
ФОРМИРОВАТЕЛЬ КОМАНДЫ ПРМ/ПРД
ИНВЕРТОР
"ПРД"
"ВКЛ"
ДЕЖ ПРИЕМ
ПРИЕМ
СХЕМА БЛОКИРОВКИ НЧ
ПЕРЕКЛЮЧАТЕЛЬ ВЫЗЫВНОЙ ЧАСТОТЫ
ПОДАВИТЕЛЬ ШУМА
ПРИЕМНИК ГЕНЕРАТОР ВЫЗОВА
ВЫЗОВ
"ПРМ"
МИКРОТЕЛЕФОН МТ-69
ТАНГЕНТА
БЛОК ПИТАНИЯ ИП2.087.196
"РАБОТА"
"ВЫХОД"
ДАТЧИК ТОКА
СХЕМА ЗАЩИТЫ
СХЕМА СРАВНЕНИЯ С ОПОРНЫМ НАПРЯЖЕНИЕМ
УСТРОЙСТВО УПРАВЛЕНИЯ ИП5.284.047
КОМПАРАТОР
ФОРМИРОВАТЕЛЬ УПРАВЛЯЮЩИХ ИМПУЛЬСОВ
16КГц
8КГц
СХЕМА УКОРОЧЕНИЯ ИМПУЛЬСА
СХЕМА ЗАПУСКА
300В
"ВКЛ."
"2,0А"
"~220В"
"ЗАЗЕМЛИТЬ"

Структурна схема системи «Естакада-1»

Додаток 2 Модернізована програма системи Matlab для здійснення FSK модуляції згідно вимог системи «Естакада-1»

```
function [y, t] = dmod_new(x, Fc, Fd, Fs, method, M,
opt2, opt3)
%DMOD
%
%WARNING: This is an obsolete function and may be removed
in the future.
%         Please use PAMMOD, QAMMOD, GENQAMMOD, FSKMOD,
PSMMOD
%         MSKMOD instead.
% $$$$$$$
M = 4;
Fc = 2638.5;
% $$$$$$$
opt_pos = 6;            % position of 1st optional parameter

if nargout > 0
   y = []; t = [];
end
% if nargin < 1
%     feval('help','dmod')
%     return;
% elseif isstr(x)
%     method = lower(deblank(x));
%     if length(method) < 3
%         error('Invalid method option for DMOD.')
%     end
%     if nargin == 1
%         % help lines for individual modulation method.
%         addition = 'See also DDEMOD, DMODCE, DDEMODCE,
MODMAP, AMOD, ADEMOD.';
%         if method(1:3) == 'qas'
%           callhelp('dmod.hlp', method(1:4), addition);
%         else
%           callhelp('dmod.hlp', method(1:3), addition);
%         end
%     else
%         % plot constellation, make a shift.
%         opt_pos = opt_pos - 3;
%         M = Fc;
%         if nargin >= opt_pos
%             opt2 = Fd;
%         else
%             modmap(method, M);
%             return;
%         end
%         if nargin >= opt_pos+1
%             opt3 = Fs;
%         else
%             modmap(method, M, opt2);
%             return;
%         end
%         modmap(method, M, opt2, opt3);  % plot
constellation
%     end
%     return;
% end

% if (nargin < 4)
%     error('Usage: Y = DMOD(X, Fc, Fd, Fs, METHOD, OPT1,
OPT2, OPT3) for passband modulation');
% elseif nargin < opt_pos-1
%     method = 'samp';
% else
%     method = lower(method);
% end

len_x = length(x);
if length(Fs) > 1
    ini_phase = Fs(2);
    Fs = Fs(1);
else
    ini_phase = 0;      % default initial phase
end

if ~isfinite(Fs) | ~isreal(Fs) | Fs<=0
    error('Fs must be a positive number.');
elseif length(Fd)~=1 | ~isfinite(Fd) | ~isreal(Fd) |
Fd<=0
    error('Fd must be a positive number.');
else
    FsDFd = Fs/Fd;      % oversampling rate
    if ceil(FsDFd) ~= FsDFd
        error('Fs/Fd must be a positive integer.');
    end
end
if length(Fc) ~= 1 | ~isfinite(Fc) | ~isreal(Fc) | Fc <=
0
    error('Fc must be a positive number. For baseband
modulation, use DMODCE.');
elseif Fs/Fc < 2
    warning('Fs/Fc must be much larger than 2 for
accurate simulation.');
end

% determine M
if isempty(findstr(method, '/arb')) &
isempty(findstr(method, '/cir'))
    if nargin < opt_pos
        M = max(max(x)) + 1;
        M = 2^(ceil(log(M)/log(2)));
        M = max(2, M);
    elseif length(M) ~= 1 | ~isfinite(M) | ~isreal(M) | M
<= 0 | ceil(M) ~= M
        error('Alphabet size M must be a positive
integer.');
    end
end

if isempty(x)
    y = [];
    return;
end
[r, c] = size(x);
if r == 1
    x = x(:);
    len_x = c;
else
    len_x = r;
end

% expand x from Fd to Fs.
if isempty(findstr(method, '/nomap'))
    if ~isreal(x) | all(ceil(x)~=x)
        error('Elements of input X must be integers in
[0, M-1].');
    end
    yy = [];
    for i = 1 : size(x, 2)
        tmp = x(:, ones(1, FsDFd)*i)';
        yy = [yy tmp(:)];
    end
    x = yy;
    clear yy tmp;
end
%***** fsk
********************************************************
*******
if strncmpi(method, 'fsk', 3)
    if nargin < opt_pos + 1
        Tone = Fd;
    else
        Tone = opt2;
    end

    if (min(min(x)) < 0)  | (max(max(x)) > (M-1))
       error('An element in input X is outside the
permitted range.');
    end

    [len_y, wid_y] = size(x);
    t = (0:1/Fs:((len_y-1)/Fs))';    % column vector with
all the time samples
    t = t(:, ones(1, wid_y));        % replicate time
vector for multi-channel operation

    %osc_freqs = pi*[-(M-1):2:(M-1)]*Tone;
    osc_freqs(1) = pi*(-1)*Tone; % для 0
    osc_freqs(2) = pi*(-2.639)*Tone; % для 1
    osc_freqs(3) = pi*Tone; % для частоти повтору
    osc_freqs(4) = pi*3*Tone; % пусто !!!@@@!!! може
замінити на 0

    osc_output = (0:1/Fs:((len_y-1)/Fs))'*osc_freqs;

    mod_phase = zeros(size(x))+ini_phase;
    for index = 1:M
        mod_phase = mod_phase +
(osc_output(:,index)*ones(1,wid_y)).*(x==index-1);
    end
    y = cos(2*pi*Fc*t+mod_phase);
```

```
%*******************************************************
******************
elseif strncmpi(method, 'samp', 4)
    % This is for converting an input signal from
sampling frequency Fd
    % to sampling frequency Fs.
    [len_y, wid_y] = size(x);
    t = (0:1/Fs:((len_y-1)/Fs))';
    y = x;
else    % invalid method
    error(sprintf(['You have used an invalid
method.\n',...
        'The method should be one of the following
strings:\n',...
        '\t''ask'' Amplitude shift keying
modulation;\n',...
        '\t''psk'' Phase shift keying modulation;\n',...
        '\t''qask'' Quadrature amplitude shift-keying
modulation, square constellation;\n',...
        '\t''qask/cir'' Quadrature amplitude shift-keying
modulation, circle constellation;\n',...
        '\t''qask/arb'' Quadrature amplitude shift-keying
modulation, user defined constellation;\n',...
        '\t''fsk'' Frequency shift keying
modulation;\n',...
        '\t''msk'' Minimum shift keying modulation.']));
end

if r==1 & ~isempty(y)
    y = y.';
end
[r, c] = size(y);
if r == 1
    y=y.';
end
% [EOF]
```

Додаток 3 Модернізована програма системи Matlab для здійснення FSK демодуляції згідно вимог системи «Естакада-1»

```
function x = ddemod_new(y, Fc, Fd, Fs, method, M, opt1,
opt2, opt3, opt4)

%WARNING: This is an obsolete function and may be removed
in the future.

M = 4;
Fc = 2638.5;

opt_pos = 7;         % position of 1st optional parameter

if nargin < 1
    feval('help', 'ddemod');
    return;
elseif isstr(y)
    method = lower(deblank(y));
    if length(method) < 3
        error('Invalid method option for ddemod.');
    end
    if nargin == 1
        addition = ['See also DMOD, AMOD, ADEMOD, DMODCE,
DDEMODCE, DEMODMAP, MODMAP,',...
                    '\r              EYEDIAGRAM,
SCATTERPLOT.'];
        addition = sprintf(addition);
        if method(1:3) == 'qas'
            callhelp('ddemod.hlp', method(1:4),
addition);
        else
            callhelp('ddemod.hlp', method(1:3),
addition);
        end
    else
        warning('Wrong number of input variables. Use
MODMAP to plot constellations.');
    end
    return;
```

```
end

if nargin < 4
    disp('Usage: Z=DDEMOD(Y, Fc, Fd, Fs, METHOD, M, OPT1,
OPT2, OPT3, OPT4) for passband demodulation');
    return;
elseif nargin < opt_pos - 2
    if nargout < 1
        method = 'eye';
    else
        method = 'sample';
    end
end
method = lower(method); % findstr is case sensitive

if length(Fs) > 1
    ini_phase = Fs(2);
    Fs = Fs(1);
else
    ini_phase = 0;      % default initial phase
end
if length(Fd) > 1
    offset = Fd(2);
    Fd = Fd(1);
else
    offset = 0;         % default timing offset
end

if ~isfinite(Fs) | ~isreal(Fs) | Fs<=0
    error('Fs must be a positive number.');
elseif ~isfinite(Fd) | ~isreal(Fd) | Fd<=0
    error('Fd must be a positive number.');
else
    FsDFd = Fs/Fd;      % oversampling rate
    if ceil(FsDFd) ~= FsDFd
        error('Fs/Fd must be a positive integer.');
    end
end
if ~isreal(offset) | ceil(offset)~=offset | offset<0 |
offset>=FsDFd
    error('OFFSET must be an integer in the range [0,
Fs/Fd).');
end
if length(Fc) ~= 1 | ~isfinite(Fc) | ~isreal(Fc) | Fc <=
0
```

```
    error('Fc must be a positive number. For baseband
demodulation, use DDEMODCE.');
elseif Fs/Fc < 2
    warning('Fs/Fc must be much larger than 2 for
accurate simulation.');
end

if (nargin >= opt_pos & isempty(findstr(method, '/arb'))
& ...
   isempty(findstr(method, '/cir')) & ...
   (length(M) ~= 1 | ~isfinite(M) | ~isreal(M) | M <= 0 |
ceil(M) ~= M))
    error('Alphabet size M must be a positive integer.');
end

if isempty(y)
    x = [];
    return;
end
[r, c] = size(y);
if r == 1
    y = y(:);
    len_y = c;
else
    len_y = r;
end
if rem(len_y, FsDFd) ~= 0
    error('Number of samples in y must be an integer
multiple of Fs/Fd.');
elseif ~isreal(y)
    error('Input Y must be real.');
end
% ***** start FSK
*******************************************************
if strncmpi(method, 'fsk', 3)
    if nargin < opt_pos
        Tone = Fd;
    else
        Tone = opt1;
    end
    if findstr(method,'/nomap')
        warning(sprintf(['The option ''/nomap'' does not
apply to FSK demodulation.\n',...
            '          The function will proceed ignoring
the ''/nomap'' switch.']));
```

```
    end

    %calculate the correlation of fsk.
    [len_y, wid_y] = size(y);
%   z = [-(M-1):2:(M-1)]  * Tone * pi / Fs;
    z(1) = (-1) * Tone * pi / Fs; % для 0
    z(2) = (-2.639) * Tone * pi / Fs; % для 1
    z(3) = Tone * pi / Fs; % для частоти повтору
    z(4) = 3*Tone * pi / Fs; % пусто !!!@@@!!! може
замінити на 0

    z = [ones(len_y, 1)]*z;
    z = cumsum(z);
    t = [0 : 1/Fs : 1/Fd-1/Fs]';
    t = t(:, ones(1, M));
    symbol_period=1/Fd;

    %leave space for x
    x = y([offset+1 : FsDFd : len_y], :);
    [len_x, wid_x] = size(x);

    if findstr(method, '/eye')
        t1 = [0 : FsDFd-1]/Fs;
        t1 = t1 + offset/Fs;
        clf;
        plot([min(t1), max(t1), max(t1)], [-1/2, NaN, 1])
        axis([min(t1) max(t1), -1/2, 1]);
        hold on
    end

    for i = 1 : wid_x
        comp_low = 1;
        if offset <= 0
            comp_upp = FsDFd;
        else
            comp_upp = offset;
        end
        for k = 1 : len_x
            if findstr(method, '/nonc')
                z_temp  = cos((t+(k-
1)*symbol_period)*2*pi*Fc + z(1:FsDFd,:));
                zz_temp = sin((t+(k-
1)*symbol_period)*2*pi*Fc + z(1:FsDFd,:));
            else
        end
    end
    if findstr(method, '/eye')
        hold off;
    end
%
***********************************************************
***************
% elseif strncmpi(method, 'msk', 3)
%   M = 2;
%   symbol_period=1/Fd;
%   t = [0 : 1/Fs : 1/Fd-1/Fs]';
%
%   if findstr(method, '/nomap')
%       warning(sprintf(['The option ''/nomap'' does not
apply to MSK demodulation.\n',...
%              '             The function will proceed ignoring
the ''/nomap'' switch.']));
%   end
%   if findstr(method,'/noncoherence')
%       warning(sprintf(['The option ''/noncoherence''
does not apply to MSK demodulation.\n',...
%              '             The function will proceed ignoring
the ''/noncoherence'' switch.']));
%   end
%   if findstr(method,'/eye')
%       warning(sprintf(['The option ''/eye'' has not
been implemented for MSK demodulation.\n',...
%              '             The function will proceed ignoring
the ''/eye'' switch.']));
%   end
%
%   %leave space for x
%   x = y([offset+1 : FsDFd : len_y], :);
%   [len_x, wid_x] = size(x);
%
%   for i = 1 : wid_x
%       comp_low = 1;
%       if offset <= 0
%           comp_upp = FsDFd;
%       else
%           comp_upp = offset;
%       end
%
%       % initial conditions for demodulator
%        sigmanminus1=0;
%        lambda0_prev=0;
%        lambda1_prev=0;
%
%        for k = 1 : len_x
%             %
%             % Based on algorithm provided by B. Rimoldi,
%             % "A Decomposition Approach to CPM," IEEE
Transactions on Information Theory,
%             % Vol. 34, No. 2, March 1988
%             %
%             % phiI and phiQ are from equations (22a) and
(22b)
%             phiI     =    sqrt(1/2)*cos(ini_phase+(t + (k-
1)*symbol_period)*2*pi*(Fc-(1/4)/symbol_period));
%             phiQ     = -1*sqrt(1/2)*sin(ini_phase+(t + (k-
1)*symbol_period)*2*pi*(Fc-(1/4)/symbol_period));
%             % s0 is determined from Figure 7 for sigman=0
and Un=0
%             % s1 is determined from Figure 7 for sigman=0
and Un=1
%             s0       =    sqrt(1/symbol_period)*phiI;
%             s1       =
sqrt(1/symbol_period)*(cos(pi*t/symbol_period).*phiI+sin(
pi*t/symbol_period).*phiQ);
%
%             if findstr(method,'/eye')
%                  % lambda0 = cumsum(y(comp_low:comp_upp,
i) .* s0(1:comp_upp-comp_low+1,:));
%                  % lambda1 = cumsum(y(comp_low:comp_upp,
i) .* s0(1:comp_upp-comp_low+1,:));
%                  % lambda0 =
lambda0/(max(max(max(abs(lambda0))),eps));
%                  % lambda1 =
lambda1/(max(max(max(abs(lambda1))),eps));
%                  % if(k==1)
%                  %   plot(t(FsDFd-
size(lambda0,1)+1:FsDFd)',lambda0, t(FsDFd-
size(lambda1,1)+1:FsDFd)',lambda1);
%                  % else
%                  %   plot(t(1:size(lambda0,1))',lambda0,
t(1:size(lambda1,1))', lambda1);
%                  % end
%                  % lambda0=lambda0(size(lambda0,1),:); %
last value
```

```
%                % lambda1=lambda1(size(lambda1,1),:); %
last value
%           else
%                % lambda0 and lambda1 are defined by (26)
for s0 and s1, respectively
%                lambda0 =    sum(y(comp_low:comp_upp, i)
.* s0(1:comp_upp-comp_low+1,:));
%                lambda1 =    sum(y(comp_low:comp_upp, i)
.* s1(1:comp_upp-comp_low+1,:));
%
%                % decision rule is based on (34)
%                if((lambda0_prev+lambda0)>(lambda1_prev-
lambda1))
%                    sigman=0;
%                else
%                    sigman=1;
%                end
%
%                lambda0_prev=lambda0;
%                lambda1_prev=lambda1;
%
%                % inverse of MSK state encoder (c.f.,
Fig. 11)
%                un=mod(sigman-sigmanminus1,2);
%                sigmanminus1=sigman;
%
%                % one symbol delay because of Viterbi
algorithm
%                if(k>1)
%                    x(k-1, i) = un;
%                end
%
%                % suboptimum decision for last symbol
%                if(k==len_x)
%                    if(lambda0>lambda1)
%                        x(k,i)=mod(0-sigmanminus1,2);
%                    else
%                        x(k,i)=mod(1-sigmanminus1,2);
%                    end
%                end
%
%                comp_low = min(comp_low + FsDFd, len_y);
%                comp_upp = min(comp_upp + FsDFd, len_y);
%           end % whether plotting eye diagram
%       end % through k symbols
%   end % through all columns of x
% elseif (strncmpi(method, 'qask', 4) | strncmpi(method,
'qam', 3) |...
%          strncmpi(method, 'qsk', 3) | strncmpi(method,
'psk', 3))
%   if findstr(method, '/ar')       % arbitrary
constellation
%       if nargin < opt_pos
%           error('Incorrect format for
METHOD=''qask/arbitrary''.');
%       end
%       I = M;
%       Q = opt1;
%         if nargin < opt_pos + 2
%             % In digital demodulation, integrator
replaced LPF.
%             num = 1;
%             den = 1;
%         else
%             num = opt2;
%           den = opt3;
%       end
%       M = length(I);
%   elseif findstr(method, '/ci')   % circular
constellation
%       if nargin < opt_pos - 1
%           error('Incorrect format for
METHOD=''qask/cir''.');
%       end
%       NIC = M;
%       M = length(NIC);
%       if nargin < opt_pos
%           AIC = [1 : M];
%       else
%           AIC = opt1;
%       end
%       if nargin < opt_pos + 1
%           PIC = NIC * 0;
%       else
%           PIC = opt2;
%       end
%         if nargin < opt_pos + 3
%             % In digital demodulation, integrator
replaced LPF.
%             num = 1;
```

```
       den = 1;
%    else
%      num = opt3;
%     den = opt4;
%   end
%   inx = apkconst(NIC, AIC, PIC);
%   I = real(inx);
%   Q = imag(inx);
%  elseif strncmpi(method, 'psk', 3)   % PSK
%    if nargin < opt_pos - 1
%      error('M-Ary number must be specified for
psk demap.');
%    end
%    NIC = M;
%    AIC = [1 : M];
%    PIC = 0;
%    if nargin < opt_pos + 1
%      num = 1;
%      den = 1;
%    else
%      num = opt1;
%      den = opt2;
%    end
%    inx = apkconst(NIC, AIC, PIC);
%   I = real(inx);
%   Q = imag(inx);
% else   % square constellation
%   [I, Q] = qaskenco(M);
%    if nargin < opt_pos + 1
%      % In digital demodulation, integrator
replaced LPF.
%      num = 1;
%      den = 1;
%    else
%      num = opt1;
%     den = opt2;
%   end
% end
%
%  % Integrate to remove double freq component and
replicate average
%  % over symbol
% y  = ademod(y, Fc, [Fs, ini_phase], 'qam', num, den);
% sizey = size(y);
% y = integ(y, FsDFd, offset);
%   y = repmat(y(:), 1, FsDFd);
%   y = reshape(y.', sizey(1), sizey(2));
%
%   if findstr(method, '/eye')
%       ddemod(y, Fc, [Fd, offset], [Fs, ini_phase],
'eye');
%   end
%   if findstr(method, '/sca')
%       ddemod(y, Fc, [Fd, offset], [Fs, ini_phase],
'sca');
%   end
%   if findstr(method, '/nomap')
%       x = y;
%   else
%       x = demodmap(y, [Fd offset], Fs, 'qask/arb', I,
Q);
%   end
elseif strncmpi(method, 'samp', 4)
    % This is for converting an input signal from
sampling frequency Fs
    % to sampling frequency Fd.
    x = demodmap(y, [Fd, offset], Fs, 'sample');
elseif strncmpi(method, 'eye', 3)
    % generate eye diagram (set offset to be the sample
of a symbol)
    eyediagram(y, FsDFd, 1, rem(offset-1+FsDFd,FsDFd));
elseif strncmpi(method, 'sca', 3)
    % generate scatterplot (set offset to be the sample
of a symbol)
    h = scatterplot(y, FsDFd, rem(offset-1+FsDFd,FsDFd));
else    % invalid method
    error(sprintf(['You have used an invalid
method.\n',...
        'The method should be one of the following
strings:\n',...
        '\t''ask'' Amplitude shift keying
modulation;\n',...
        '\t''psk'' Phase shift keying modulation;\n',...
        '\t''qask'' Quadrature amplitude shift-keying
modulation, square constellation;\n',...
        '\t''qask/cir'' Quadrature amplitude shift-keying
modulation, circle constellation;\n',...
        '\t''qask/arb'' Quadrature amplitude shift-keying
modulation, user defined constellation;\n',...
```

```
        '\t''fsk'' Frequency shift keying
modulation;\n',...
        '\t''msk'' Minimum shift keying
modulation;\n',...
        '\t''sample'' Convert sample frequency Fs input
to sample frequency Fd output.']));
end

if r==1 & ~isempty(x)
    x = x.';
end

%----------------------------------
function y = integ(x, osr, offset)
%INTEG Integrator.
%   INTEG integrates the analog demodulated signal x for
1 symbol period,
%   then output 1 value into y. osr is the oversampling
rate (number of
%   samples for 1 symbol). offset is the timing offset
(starting point of
%   integration).

[xRow, xCol] = size(x);

% Shift x upward due to timing offset
x = [x((offset+1):end, :); zeros(offset, xCol)];

% Integration & dump = taking mean value of samples of
each symbol
x = mean(reshape(x, osr, xRow*xCol/osr), 1);

y = reshape(x, xRow/osr, xCol);

% [EOF]
```