15 Make it So

It always seems impossible until it's done.

Nelson Mandela, President of South Africa, Nobel Prize for Peace 1993

This chapter describes a software-defined radio design project called $\mathcal{M}^6$, the *Mix 'n' Match Mostly Marvelous Message Machine*[1]. The $\mathcal{M}^6$ transmission standard is specified so that the receiver can be designed using the building blocks of the preceding chapters. The DSP portion of the $\mathcal{M}^6$ can be simulated in MATLAB by combining the functions and subroutines from the examples and exercises of the previous chapters.

The input to the digital portion of the $\mathcal{M}^6$ receiver is a sampled signal at intermediate frequency (IF) that contains several simultaneous messages, each transmitted in its own frequency band. The original message is text that has been converted into symbols drawn from a 4-PAM constellation, and the pulse shape is a square-root raised cosine. The sample frequency can be less than twice the highest frequency in the analog IF signal, but it must be sufficiently greater than the inverse of the transmitted symbol period to be twice the bandwidth of the baseband signal. The successful $\mathcal{M}^6$ MATLAB program will demodulate, synchronize, equalize, and decode the signal, so it is a "fully operational" software-defined receiver (although it is not intended to work in "real time"). The receiver must overcome multiple impairments. There may be phase noise in the transmitter oscillator. There may be an offset between the frequency of the oscillator in the transmitter and the frequency of the oscillator in the receiver. The pulse clocks in the transmitter and receiver may differ. The transmission channel may be noisy. Other users in spectrally adjacent bands may be actively transmitting at the same time. There may be intersymbol interference caused by multipath channels.

The next section describes the transmitter, the channel, and the analog front-end of the receiver. Then Section 15.2 makes several generic observations about receiver design, and proposes a methodology for the digital receiver design. The

[1] This version of Chapter 15 has been edited to include a TDMA source and is used for the WWU EE 460 Spring 2020 final project.

final section describes the receiver design challenge that serves as the culminating design experience of this book. Actually building the $\mathcal{M}^6$ receiver, however, is left to you. You will know that your receiver works when you can recover the mystery message hidden inside the received signal.

15.1 How the Received Signal Is Constructed

Receivers cannot be designed in a vacuum; they must work in tandem with a particular transmitter. Sometimes, a communication system designer gets to design both ends of the system. More often, however, the designer works on one end or the other, with the goal of making the signal in the middle meet some standard specifications. The standard for the $\mathcal{M}^6$ is established on the transmitted signal, and consists, in part, of specifications on the allowable bandwidth and on the precision of its carrier frequency. The standard also specifies the source constellation, the modulation, and the coding schemes to be used. The front-end of the receiver provides some bandpass filtering, downconversion to IF, and automatic gain control prior to the sampler.

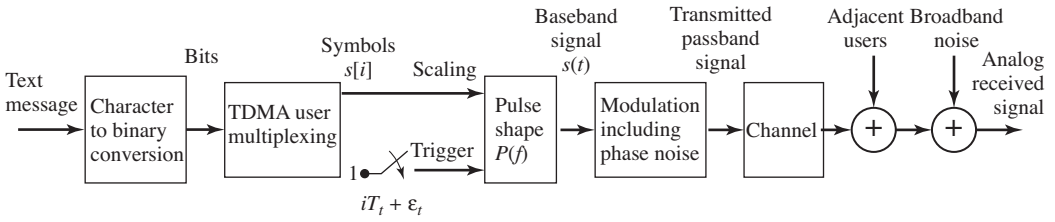

Figure 15.1 Received signal generator.

The $\mathcal{M}^6$ transmitter is a multiuser transmitter where the multiple access capability is enabled with time-division multiple access (TDMA). The scenario we consider here is the downlink in cellular telephony, which was also considered in Homework 1. The transmitter (i.e. the base station) transmits to 3 users simultaneously, and the receiver (i.e. the mobile device) retains only its intended portion of the signal.

This section describes the construction of the sampled IF signal that must be processed by the $\mathcal{M}^6$ receiver. The system that generates the analog received signal is shown in block diagram form in Figure 15.1. The front end of the receiver that turns this into a sampled IF signal is shown in Figure 15.2.

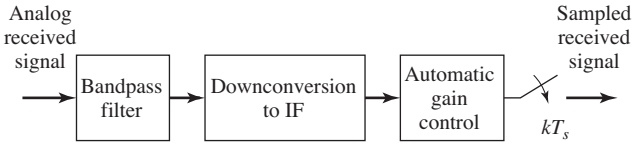

Figure 15.2 Front end of the receiver.

The original message in Figure 15.1 is a character string of English text. Each character is converted into a 7-bit binary string according to the ASCII conversion format (e.g., the letter "a" is 1100001 and the letter "M" is 1001101), as in Example 8.2. Note that this is different from the 8-bit ASCII encoding that is used in `letters2pam.m`. The resulting bit string is then coded using the (5,2) linear block code specified in Section 14.6.1 which adds redundancy into the bits by associating a 5-bit code with each pair of bits. The output of the block code is then partitioned into pairs that are associated with the four integers of a 4-PAM alphabet ± 1 and ± 3 via the mapping

$$\begin{aligned} 11 &\rightarrow +3 \\ 10 &\rightarrow +1 \\ 01 &\rightarrow -1 \\ 00 &\rightarrow -3 \end{aligned} \tag{15.1}$$

as in Example 8.1. Thus, if there are n letters, there are $7n$ (uncoded) bits, $7n(\frac{5}{2})$ coded bits, and $7n(\frac{5}{2})(\frac{1}{2})$ 4-PAM symbols. These mappings are familiar from Section 8.1, and Exercise 14.25 provides more details about the $\mathcal{M}^6$ encoding. A new and improved pair of MATLAB functions have been provided on the course website called `letters2pam2.m` and `pam2letters2.m` which convert between a text string and a sequence of 4-PAM symbols $s[i]$ according to this encoding specification.

In order to decode the message at the receiver, the recovered symbols must be properly grouped, the start of each group must be located, and the desired user data must be extracted. To aid this TDMA frame synchronization, a preamble is inserted in the symbol stream at the start of every frame. The preamble sequence that starts each frame is given by the phrase

$$\text{A0Oh well whatever Nevermind} \tag{15.2}$$

which codes into 245 4-PAM symbols and is assumed to be known at the receiver. This preamble can be used as a training sequence by the adaptive equalizer. The unknown messages for each user begin immediately after the preamble, and this structure is depicted in Fig. 15.3. As shown, the $\mathcal{M}^6$ symbol stream consists of coded messages periodically interrupted by the same preamble clump.

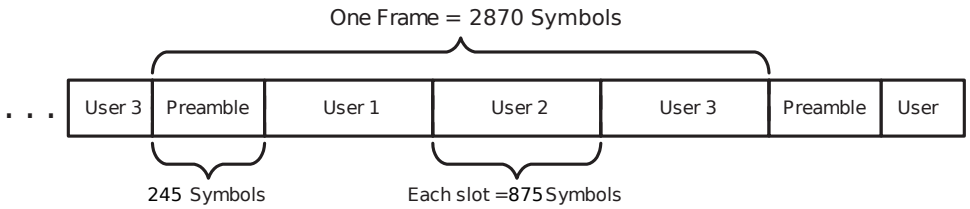

Figure 15.3 TDMA frame structure

As indicated in Figure 15.1, pulses are initiated at intervals of T_t seconds, and each is scaled by the 4-PAM symbol value. This translates the discrete-time symbol sequence $s[i]$ (composed of the coded message interleaved with the preamble) into a continuous-time signal

$$s(t) = \sum_i s[i]\delta(t - iT_t - \epsilon_t).$$

The actual transmitter symbol period T_t is required to be within 0.01 percent of a nominal $\mathcal{M}^6$ symbol period $T = 6.4$ microseconds. The transmitter symbol period clock is assumed to be steady enough that the timing offset ϵ_t and its period T_t are effectively time-invariant over the duration of a single frame.

Details of the $\mathcal{M}^6$ transmission specifications are given in Table 15.1. The pulse-shaping filter $P(f)$ is a square-root raised cosine filter symmetrically truncated to eight symbol periods. The rolloff factor β of the pulse-shaping filter is fixed within some range and is known at the receiver, though it could take on different values with different transmissions. The (half-power) bandwidth of the square-root raised cosine pulse could be as large as ≈102 kHz for the nominal T. With double sideband modulation, the pulse shape bandwidth doubles so that each passband FDM signal will need a bandwidth at least 204 kHz wide.

The channel may be near ideal (i.e., a unit gain multisymbol delay) or it may have significant intersymbol interference. In either case, the impulse response of the channel is unknown at the receiver, though an upper bound on its delay spread may be available. There are also disturbances that may occur during the transmission. These may be wideband noise with flat power spectral density or they may be narrowband interferers, or both. They are unknown at the receiver.

The achieved intermediate frequency is required to be within 0.01 percent of its assigned value. The carrier phase $\theta(t)$ is unknown to the receiver and may vary over time, albeit slowly. This means that the phase of the intermediate frequency signal presented to the receiver sampler may also vary.

The bandpass filter before the downconverter in the front-end of the receiver in Figure 15.2 partially attenuates adjacent 204 kHz wide FDM user bands. The automatic gain control is presumed locked and fixed over each transmission. The free-running sampler frequency of 850 kHz is well above twice the 102 kHz baseband bandwidth of the user of interest. This is necessary for the baseband analog signal interpolator used in the timer in the DSP portion of the receiver in Figure 15.4. However, the sampler frequency is not above twice the highest frequency of the IF signal. This means that the sampled received signal has replicated the spectrum at the output of the front-end analog downconverter lowpass filter to frequencies between zero and IF.

Table 15.1. $\mathcal{M}^6$ Transmitter Specifications

Symbol source alphabet	$\pm 1, \pm 3$
Assigned intermediate frequency	2 MHz
Nominal symbol period	6.4 microseconds
SRRC pulse shape rolloff factor	$\beta \in [0.1, 0.3]$
FDM user slot allotment	204 kHz
Width of SRRC pulse shape	8 clock periods
Preamble sequence	See (15.2)
Preamble length	245 symbols
Data length	875 symbols/user
Number of users	3 users
Total frame length	2870 symbols
Time-varying IF carrier phase	Filtered white noise
IF offset	Fixed, $< 0.01\%$
Timing offset	Fixed
Symbol period offset	Fixed, $< 0.01\%$
Intersymbol interference	Max. delay spread = 7 symbols
Sampler frequency	850 kHz

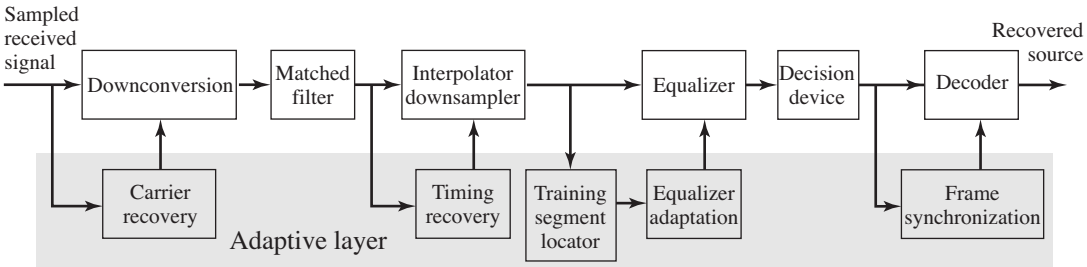

Figure 15.4 DSP portion of software-defined receiver.

15.2 A Design Methodology for the $\mathcal{M}^6$ Receiver

Before describing the specific design requirements that must be met by a successful $\mathcal{M}^6$ receiver, this section makes some generic remarks about a systematic approach to receiver design. There are four generic stages:

1. *Choose the order in which the basic operations of the receiver occur.*
2. *Select components and methods that can perform the basic operations in an ideal setting.*
3. *Select adaptive elements that allow the receiver to continue functioning when there are impairments.*
4. *Verify that the performance requirements are met.*

While it may seem as though each stage requires that choices made in the preceding stages be fixed, in reality, difficulties encountered at one stage in the design

process may require a return to (and different choices to be made in) earlier stages. As will soon become clear, the $\mathcal{M}^6$ problem specification has basically (pre)resolved the design issues of the first two stages.

15.2.1 Stage One: Ordering the Pieces

The first stage is to select the basic components and the order in which they occur. The design layout first established in Figure 2.13 (and reappearing in the schematic of the DSP portion of the receiver in Figure 15.4) suggests one feasible structure. As the signal enters the receiver it is downconverted (with carrier recovery), matched filtered, interpolated (with timing recovery), equalized (adaptively), quantized, and decoded (with frame synchronization). This classical ordering, while popular, is not the only (nor necessarily the best) way to recover the message from the noisy, ISI-distorted, FDM–PAM–IF received signal. However, it offers a useful foundation for assessing the relative benefits and costs of alternative receiver configurations. Also, we know for sure that the $\mathcal{M}^6$ receiver can be built this way. Other configurations may work, but we have not tested them.[2]

How was this ordering of components chosen? The authors have consulted with, worked for, talked about (and argued with) engineers working on a number of receiver systems including HDTV (high definition television), DSL, and AlohaNet. The ordering of components in Figures 2.13 and 15.4 represents an amalgamation of ideas from these (and other) systems. Sometimes it is easy to argue why a particular order is good, sometimes it is a matter of preference or personal experience, and sometimes the choice is based on factors outside the engineer's control.[3]

For example, the carrier recovery algorithms of Chapter 10 are not greatly affected by noise or intersymbol interference (as was shown in Exercises 10.36 and 10.40). Thus carrier recovery can be done before equalization, and this is the path we have followed. But it need not be done in this order.[4] Another example is the placement of the timing recovery element. The algorithms of Chapter 12 operate at baseband, and hence the timing recovery in Figure 15.4 is placed after the demodulation. But there are passband timing recovery algorithms that could have been used to reverse the order of these two operations.

[2] If this sounds like a challenge, rest assured it is. Research continues worldwide, making compilation of a complete handbook of receiver designs and algorithms a Sisyphean task. The creation of "new" algorithms with minor variations that exploit a particular application-specific circumstance is a popular pastime of communication engineers. Perhaps you too will come up with a unique approach!

[3] For instance, the company might have a patent on a particular method of timing recovery and using any other method might require royalty payments.

[4] For instance, in the QAM radio of *A Digital Quadrature Amplitude Modulation Radio*, available in the website, the blocks appear in a different order.

15.2.2 Stage Two: Selecting Components

Choices for the second design stage are relatively set as well. Since the sampling is done at a sub-Nyquist rate f_s (relative to the IF frequency f_I), the spectrum of the analog received signal is replicated every f_s. The integer n for which $f^\dagger = |f_I - nf_s|$ is smallest defines the nominal frequency $f^\dagger$ from which further downconversion is needed. Recall that such downconversion by sampling was discussed in Section 6.2. Using different specifications, the $\mathcal{M}^6$ sampling frequency f_s may be above the Nyquist frequency associated with the IF frequency f_I.[5]

The most common method of downconversion is to use mixing followed by an FIR lowpass filter. This will be followed by an FIR matched filter, an interpolator–decimator for downsampling, and a symbol-spaced FIR equalizer that adapts its coefficients based on the training data contained in the transmission. The output of the equalizer is quantized to the nearest 4-PAM symbol value, translated back into binary, decoded (using the (5,2) block decoder) and finally turned back into readable text.

Given adequate knowledge of the operating environment (the SNR in the received signal, the carrier frequency and phase, the clock period and symbol timing, and the preamble location), the designer-selected parameters within these components can be set to recover the message. This was, in fact, the strategy followed in the idealized receiver of Chapter 9. Said another way, the choices in stages one and two are presumed to admit an acceptable answer if properly tuned. Component selections at this point (including specification of the fixed lowpass filter in the downconverter and the fixed matched filter preceding the interpolator/downsampler) can be confirmed by simulations of the ISI-free ideal/full-knowledge setting. Thus, the upper half of Figure 15.4 is specified by stage two activities.

15.2.3 Stage Three: Anticipating Impairments

In the third design stage, the choices are less constrained. Elements of the third stage are shown in the lower half of the receiver schematic (the "adaptive layer" of Figure 15.4) and include the selection of algorithms for carrier, timing, frame synchronization, and equalizer adaptation. There are several issues to consider.

One of the primary stage three activities is algorithm selection—which performance function to use in each block. For example, should the $\mathcal{M}^6$ receiver use a phase-locked loop, a Costas loop, or a decision-directed method for carrier recovery? Is a dual loop needed to provide adequate carrier tracking, or will a single loop suffice? What performance function should be used for the equalizer? Which algorithm is best for the timing recovery? Is simple correlation suitable to locate the preamble segment?

[5] Indeed, changing parameters such as this allows an instructor to create new transmission "standards" for each class!

Once the specific methods have been chosen, it is necessary to select specific variables and parameters within the algorithms. This is a traditional aspect of engineering design that is increasingly dominated by computer-aided design, simulation, and visualization tools. For example, error surfaces and eye diagrams can be used to compare the performance of the various algorithms in particular settings. They can be used to help determine which technique is more effective for the application at hand.

As software-aided design packages proliferate, the need to understand the computational mechanics underlying a particular design becomes less of a barrier. For instance, **Software Receiver Design** has relied exclusively on the filter design algorithms built into MATLAB. But the specification of the filter (its shape, cutoff frequencies, computational complexity, and filter length) cannot be left to MATLAB. The more esoteric the algorithm, the less transparent is the process of selecting design parameters. Thus, **Software Receiver Design** has devoted considerable space to the design and operation of adaptive elements.

But, even assuming that the tradeoffs associated with each of the individual components are clear, how can everything be integrated together to succeed at a multifaceted design objective such as the $\mathcal{M}^6$ receiver?

15.2.4 Sources of Error and Tradeoffs

Even when a receiver is fully operational, it may not decode every symbol precisely. There is always a chance of error. Perhaps part of the error is due to a frequency mismatch, part of the error is due to noise in the channel, part of the error is due to a nonoptimal timing offset, etc. This section (and the next) suggest a general strategy for allocating "part of" the error to each component. Then, as long as the sum of all the partial errors does not exceed the maximum allowable error, there is a good chance that the complete receiver will work according to its specifications.

The approach is to choose a method of measuring the amount of error, for instance, the average of the squared recovery error. Each individual component can be assigned a threshold, and its parameters can be adjusted so that it does not contribute more than its share to the total error. Assuming that the accumulation of the errors from various sources is additive, the complete receiver will have no larger error than the concatenation of all its parts. This additivity assumption is effectively an assumption that the individual pieces of the system do not interact with each other. If they do (or when they do), then the threshold allotments may need to be adjusted.

There are many factors that contribute to the recovery error, including the following:

- Residual interference from adjacent FDM bands (caused by imperfect bandpass filtering before downconversion and imperfect lowpass filtering after downconversion).

- AGC jitter (caused by the deviation in the instantaneous signal from its desired average and scaled by the stepsize in the AGC element).
- Quantization noise in the sampler (caused by coarseness in the magnitudes of the quantizer).
- Round-off noise in filters (caused by wordlength limitations in filter parameters and filter algebra).
- Residual interference from the doubly upconverted spectrum (caused by imperfect lowpass filtering after downconversion).
- Carrier phase jitter (occurs physically as a system impairment and is caused by the stepsize in the carrier recovery element).
- Timing jitter (occurs physically as a system impairment and is caused by the stepsize in the timing recovery element).
- Residual mean squared error left by the equalizer (even an infinitely long linear equalizer cannot remove all recovery error in the presence of simultaneous channel noise and ISI).
- Equalizer parameter jitter (caused by the nonvanishing stepsize in the adaptive equalizer).
- Noise enhancement by the equalizer (caused by ISI that requires large equalizer gains, such as a deep channel null at frequencies that also include noise).

Because MATLAB implements all calculations in floating point arithmetic, the quantization and round-off noise in the simulations is imperceptible. The project setup presumes that the AGC has no jitter. A well-designed and sufficiently long lowpass filter in the downconverter can effectively remove the interference from outside the user band of interest. The in-band interference from sloppy adjacent FDM signals should be considered part of the in-band channel noise. This leaves carrier phase, timing jitter, imperfections in the equalizer, tap jitter, and noise gain. All of these are potentially present in the $\mathcal{M}^6$ software-defined digital radio.

In all of the cases in which error is due to the jiggling of the parameters in adaptive elements (in the estimation of the sampling instants, the phase errors, the equalizer taps), the errors are proportional to the stepsize used in the algorithm. Thus, the (asymptotic) recovery error can be made arbitrarily small by reducing the appropriate stepsize. The problem is that, if the stepsize is too small, the element takes longer to converge. If the time to convergence of the element is too long (for instance, longer than the complete message), then the error is increased. Accordingly, there is some optimal stepsize that is large enough to allow rapid convergence yet small enough to allow acceptable error. An analogous trade-off arises with the choice of the length of the equalizer. Increasing its length reduces the size of the residual error. But as the length grows, so does the amount of tap jitter.

Such tradeoffs are common in any engineering design task. The next section suggests a method of quantifying the tradeoffs to help make concrete decisions.

15.2.5 Tuning and Testing

The testing and verification stage of receiver design is not a simple matter because there are so many things that can go wrong. (There is so much stuff that can happen!) Of course, it is always possible to simply build a prototype and then test to see if the specifications are met. Such a haphazard approach may result in a working receiver, but then again, it may not. Surely there is a better way! This section suggests a commonsense approach that is not uncommon among practicing engineers. It represents a "practical" compromise between excessive analysis (such as one might find in some advanced communication texts) and excessive trial and error (such as "try something and cross your fingers").

The idea is to construct a simulator that can create a variety of test signals that fall within the $\mathcal{M}^6$ specification. The parameters within the simulator can then be changed one at a time, and their effect noted on various candidate receivers. By systematically varying the test signals, the worst components of the receiver can be identified and then replaced. As the tests proceed, the receiver gradually improves. As long as the complete set of test signals accurately represents the range of situations that will be encountered in operation, the testing will lead to a successful design.

Given the particular stage one and two design choices for the $\mathcal{M}^6$ receiver, the previous section outlined the factors that may degrade the performance of the receiver. The following steps suggest some detailed tests that may facilitate the design process:

- Step 1: *Tuning the Carrier Recovery* As shown in Chapter 10, any of the carrier recovery algorithms are capable of locating a fixed phase offset in a receiver in which everything else is operating optimally. Even when there is noise or ISI, the best settings for the frequency and phase of the demodulation sinusoid are those that match the frequency and phase of the carrier of the IF signal. For the $\mathcal{M}^6$ receiver, there are two issues that must be considered. First, the $\mathcal{M}^6$ specification allows the frequency to be (somewhat) different from its nominal value. Is a dual-loop structure needed? Or can a single loop adequately track the expected variations? Second, the transmitter phase may be jittering.

  The user choosable features of the carrier recovery algorithms are the LPF and the algorithm stepsize, both of which influence the speed at which the estimates can change. Since the carrier recovery scheme needs to track a time-varying phase, the stepsize cannot be chosen too small. Since a large stepsize increases the error due to phase jitter, it cannot be chosen too large. Thus, an acceptable stepsize will represent a compromise.

  To conduct a test to determine the stepsize (and LPF) requires creating test signals that have a variety of off-nominal frequency offsets and phase jitters. A simple way to model phase jitter is to add a lowpass filtered version of zero-mean white noise to a nominal value. The quality of a particular set of parameters can then be measured by averaging (over all the test signals) the

mean squared recovery error. Choosing the LPF and stepsize parameters to make this error as small as possible gives the "best" values. This average error provides a measure of the portion of the total error that is due to the carrier recovery component in the receiver.

- Step 2: *Tuning the Timing Recovery* As shown in Chapter 12, there are several algorithms that can be used to find the best timing instants in the ideal setting. When the channel impairment consists purely of additive noise, the optimal sampling times remain unchanged, though the estimates will likely be more noisy. As shown by Example 12.3, and in Figure 12.12, however, when the channel contains ISI, the answer returned by the algorithms differs from what might be naively expected.

  There are two parts to the experiments at this step. The first is to locate the best timing recovery parameter for each test signal. (This value will be needed in the next step to assess the performance of the equalizer.) The second part is to find the mean squared recovery error due to jitter of the timing recovery algorithm.

  The first part is easy. For each test signal, run the chosen timing recovery algorithm until it converges. The convergent value gives the timing offset (and indirectly specifies the ISI) to which the equalizer will need to respond. (If it jiggles excessively, then decrease the stepsize.)

  Assessing the mean squared recovery error due to timing jitter can be done much like the measurement of jitter for the carrier recovery: measure the average error that occurs over each test signal when the algorithm is initialized at its optimum answer; then average over all the test signals. The answer may be affected by the various parameters of the algorithm: the δ that determines the approximation to the derivative, the `l` parameter that specifies the time support of the interpolation, and the stepsize (these variable names are from the first part of the timing recovery algorithm `clockrecDD.m` on page 259.)

  In operation, there may also be slight inaccuracies in the specification of the clock period. When the clock period at the transmitter and receiver differ, the stepsize must be large enough so that the timing estimates can follow the changing period. (Recall the discussion surrounding Example 12.4.) Thus, again, there is a tension between a large stepsize needed to track rapid changes and a small stepsize to minimize the effect of the jitter on the mean squared recovery error. In a more complex environment, in which clock phases might be varying, it might be necessary to follow a procedure more like that considered in step 1.
- Step 3: *Tuning the Equalizer* After choosing the equalizer method (as specified by the performance function), there are a number of parameters that must be chosen and decisions that must be made in order to implement the linear equalizer. These are
  - the order of the equalizer (number of taps),
  - the initial values of the equalizer,
  - the training signal delay (if using the training signal), and

– the stepsize.

As in the previous steps, it is a good idea to create a collection of test signals using a simulation of the transmitter. To test the performance of the equalizer, the test signals should contain a variety of ISI channels and/or additive interferences.

As suggested in Chapter 13, in a high SNR scenario the T-spaced equalizer tries to implement an approximation of the inverse of the ISI channel. If the channel is mild, with all its roots well away from the unit circle, then its inverse may be fairly short. But if the channel has zeros that are near the unit circle, then its FIR inverse may need to be quite long. While much can be said about this, a conservative guideline is that the equalizer should be from two to five times longer than the maximum anticipated channel delay spread. One subtlety that arises in making this decision and in consequent testing is that any channel ISI that is added into a simulation may appear differently at the receiver because of the sampling. This effect was discussed at length in Section 13.1, where it was shown how the effective digital model of the channel includes the timing offset. Thus (as mentioned in the previous step) assessing the "actual" channel to which the equalizer will adapt requires knowing the timing offset that will be found by the timing recovery. Fortunately, in the $\mathcal{M}^6$ receiver structure of Figure 15.4, the timing recovery algorithm operates independently of the equalizer, and so the optimal value can be assessed beforehand.

For most of the adaptive equalizers in Chapter 13, the center spike initialization is used. This was justified in Section 13.4 (see page 291) as a useful method of initialization. Only if there is some concrete a prior knowledge of the channel characteristics would other initializations be used.

The problem of finding an appropriate delay was discussed in Section 13.2.3, where the least squares solution was recomputed for each possible delay. The delay with the smallest error was the best. In a real receiver, it will not be possible to do an extensive search, and so it is necessary to pick some delay. The $\mathcal{M}^6$ receiver uses correlation to locate the preamble sequence and this can be used to locate the time index corresponding to the first training symbol. This location plus half the length of the equalizer should correspond closely to the desired delay. Of course, this value may change depending on the particular ISI (and channel lengths) used in a given test signal. Choose a value that, over the complete set of test signals, provides a reasonable answer.

The remaining designer-selected variable is stepsize. As with all adaptive methods, there is a tradeoff inherent in stepsize selection: making it too large can result in excessive jitter or algorithm instability, while making it too small can lead to an unacceptably long convergence time. A common technique is to select the largest stepsize consistent with achievement of the component's assigned asymptotic performance threshold.

- Step 4: *Frame Synchronization* Any error in locating the first symbol of each four-symbol block can completely garble the reconstructed text. The frame

synchronizer operates on the output of the quantizer, which should contain few errors once the equalizer, timing recovery, and phase recovery have converged. The success of frame synchronization relies on the peakiness of the correlation of the preamble sequence. The chosen preamble "A0Oh well whatever Nevermind" should be long enough that there are few false spikes when correlating to find the start of the message within each block. To test software written to locate the preamble, feed it a sample symbol string assembled according to the specifications described in the previous section as if the downconverter, clock timing, equalizer, and quantizer had recovered the transmitted symbol sequence perfectly.

Finally, after tuning each component separately, it is necessary to confirm that when all the pieces of the system are operating simultaneously, there are no excessive negative interactions. Hopefully, little further tuning will prove necessary to complete a successful design. The next section has more specifics about the $\mathcal{M}^6$ receiver design.

Exercise 15.1. From the parameters in Table 15.1,

a. Determine the oversampling (or upsampling) factor.
b. Determine what frequency the sampled received signal will be centered around. Recall that the receiver frontend samples at a rate lower than the IF frequency, and your answer should be something between 0 and $f_s/2$.

Exercise 15.2. The B^3IG Transmitter can be used to implement the $\mathcal{M}^6$ transmission specifications, as provided in `m6params.m`. Plot the received signal and its spectrum. Use these plots to explain why your answers to Exercise 15.1 are correct.

15.3 No Soap Radio: The $\mathcal{M}^6$ Receiver Design Challenge

The analog front end of the receiver in Figure 15.2 takes the signal from an antenna, amplifies it, and crudely bandpass filters it to (partially) suppress frequencies outside the desired user's frequency band. An analog downconverter modulates the received signal (approximately) down to the nominal intermediate frequency f_I at 2 MHz. The output of the analog downconverter is set by an automatic gain controller to fit the range of the sampler. The output of the AGC is sampled at a rate of $1/T_s = 850$ kHz to give $r[k]$, which provides a "Nyquist" bandwidth of 425 kHz that is ample for a 102 kHz baseband user bandwidth. The sampled received signal $r[k]$ from Figure 15.2 is the input to the DSP portion of the receiver in Figure 15.4.

The following comments on the components of the digital receiver in Figure 15.4 help characterize the design task:

- The downconversion to baseband uses the sampler frequency f_s, the known intermediate frequency f_I, and the current phase estimates to determine the mixer frequency needed to demodulate the signal. The $\mathcal{M}^6$ receiver may use any of the phase tracking algorithms of Chapter 10. A second loop may also help with frequency offset.
- The lowpass filtering in the demodulator should have a bandwidth of roughly 102 kHz, which will cover the selected source spectrum but reject components outside the frequency band of the desired user.
- The interpolator/downsampler implements the reduction in sample rate to T-spaced values. This block must also implement the timing synchronization, so that the time between samples after timing recovery is representative of the true spacing of the samples at the transmitter. You are free to implement this in any of the ways discussed in Chapter 12.
- Since there could be a significant amount of intersymbol interference due to channel dynamics, an equalizer is essential. Any one will do. A trained equalizer requires finding the start of the preamble while a blind equalizer may converge more slowly.
- The decision device is a quantizer defined to reproduce the known alphabet of the $s[i]$ by a memoryless nearest-element decision.
- At the final step, the decoding from 4-PAM and block decoding which removes redundancy can be accomplished with `pam2letters2.m`, thereby reconstructing the original text. This also requires a frame synchronization that finds and removes the start block consisting of the preamble, which is most likely implemented using a correlation technique. Finally, the desired user needs to be extracted from the frame.

The software-defined radio should have the following user-selectable variables that can readily be set at the start of processing of the received block of data:

- rolloff factor β for the square-root raised cosine pulse shape,
- initial phase offset,
- initial timing offset, and
- initial equalizer parameterization.

The following are some suggestions:

- Use the B^3IG Transmitter with `m6params.m` to implement the $\mathcal{M}^6$ transmission specification along with various impairments. This will enable you to test your receiver as described in the methodology proposed in the preceding section over a wider range of conditions than just the test signals available on the website.
- Try to break your receiver. See how much noise can be present in the received signal before accurate (e.g., less than 1% symbol errors) demodulation seems impossible. Find the fastest change in the carrier phase that your receiver can track, even with a bad initial guess.

- In order to facilitate more effective debugging while building the project, implementation of a debug mode in the receiver is recommended. The information of interest will be plots of the time histories of pertinent signals as well as timing information (e.g., a graph of matched filter average output power versus receiver symbol timing offset). One convenient way to add this feature to your MATLAB receiver would be to include a debug flag as an argument that produces these plots when the flag is activated.
- When debugging adaptive components, use a test with initialization at the right answer and zero stepsize to determine if the problem is in the adaptation or in the fixed component structure. An initialization very near the desired answer with a small stepsize will reveal that the adaptive portion is working properly if the adaptive parameter trajectory remains in the close vicinity of the desired answer. A rapid divergence may indicate that the update has the wrong sign or that the stepsize is way too large. An aimless wandering that drifts away from the vicinity of the desired answer represents a more subtle problem that requires reconsideration of the algorithm code and/or its suitability for the circumstance at hand.

Several test files that contain a "mystery signal" are available on the website. They are labelled `easy.mat`, `medium.mat`, and `hard.mat`. These have been created with a variety of different rolloff factors, carrier frequencies, phase noises, ISI, interferers, and symbol timing offsets. We encourage the adventurous reader to try to "receive" these secret signals. Solve the mystery. Break it down.

For Further Reading

An overview of a practical application of *software-defined radio* emphasizing the redefinability of the DSP portion of the receiver can be found in

- B. Bing and N. Jayant, "A cellphone for all standards," *IEEE Spectrum*, pg 34–39, May 2002.

The field of "software radio" erupted with a special issue of the *IEEE Communications Magazine* in May 1995. This was called a "landmark special issue" in an editorial in the more recent

- J. Mitola, III, V. Bose, B. M. Leiner, T. Turletti and D. Tennenhouse, Ed., *IEEE Journal on Selected Areas in Communications* (Special Issue on Software Radios), vol. 17, April 1999.

For more information on the technological context and the relevance of software implementations of communications systems, see

- E. Buracchini, "The Software Radio Concept," *IEEE Communications Magazine*, vol. 38, pp. 138–143, September 2000

and papers from the (occasional) special section in the *IEEE Communications Magazine* on topics in software and DSP in radio. For much more, see

- J. H. Reed, *Software Radio: A Modern Approach to Radio Engineering*, Prentice Hall, 2002

which overlaps in content (if not style) with the first half of **Software Receiver Design**.

Two recommended monographs that include more attention than most to the methodology of the same slice of digital receiver design as we consider here are

- J. A. C. Bingham, *The Theory and Practice of Modem Design*, Wiley Interscience, 1988. (especially Chapter 5)
- H. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*, Wiley Interscience, 1998 (especially Section 4.1).