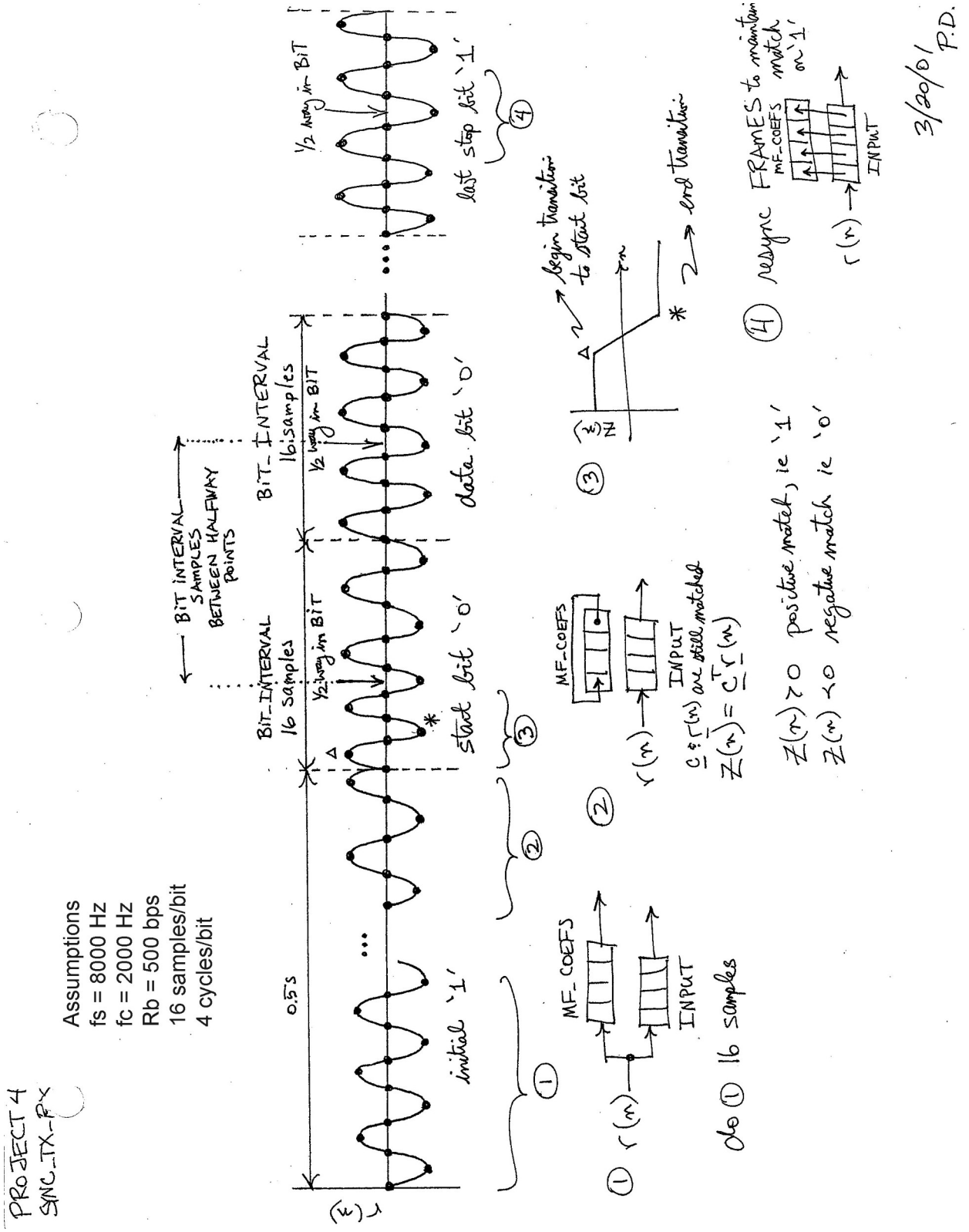
Project #4: Binary Phase-Shift Keying Modem (Receiver)



Receiver Algorithm

- Assume transmitter is sending waveform for '1' for 5.0s
- See user_data.h file for additional specifications
- SAMP_COUNTER (tracks which sample we are on inside a bit)
- BIT_COUNTER (tracks which bit we are on inside a frame)

```
INITIALIZE
      Set STARTUP_FLAG
      STARTUP_COUNTER = SAMPS_BIT, BIT_COUNTER = 11

MAIN1 (For each sample period)
      If STARTUP_FLAG
            STARTUP
      Elseif WAIT_FLAG
            WAIT_FOR_FIRST_START_BIT
      Elseif ADVANCE_FLAG
            ADVANCE_HALFWAY_THROUGH_BIT
      Else DETERMINE_FLAG
            UPDATE_VECTORS
            DETERMINE_BIT (Detect first start bit of first frame)
            If (BIT == '1') (synchronization failure—START_BIT should've been '0')
                  STOP
            Else (synchronization success)
                  Goto MAIN2
            End
      End
      Goto MAIN1

MAIN2 (For each sample period)
      UPDATE_VECTORS
      Decrement SAMP_COUNTER
      If (SAMP_COUNTER == 0)
            DETERMINE_BIT
      End
      If (BIT_COUNTER == 0)
            PROCESS_FRAME
            BIT_COUNTER = 11
            RESYNCHRONIZE
      End
      Goto MAIN2
```

```
STARTUP (Assume transmitter is sending waveform for BIT = '1')
      Shift samples into MF_COEFS and INPUT vectors
      Decrement STARTUP_COUNTER
      If STARTUP_COUNTER == 0
            Clear STARTUP_FLAG, set WAIT_FLAG
      End
```

```
WAIT_FOR_FIRST_START_BIT
      UPDATE_VECTORS
      COMPUTE_FILTER_OUTPUT
      If z < -THRESHOLD (transition detected—hopefully 1st start bit)
            Clear WAIT_FLAG, set ADVANCE_FLAG
            ADVANCE_COUNTER = SAMPS_BIT/2 − N + 1
```

```
      End
```

ADVANCE_HALFWAY_THROUGH_BIT_INTERVAL

```
(By the time z < -THRESHOLD, we're already N–1 samples into 1st start bit, so
SAMPS_BIT/2–N+1 more samples to move halfway into bit interval.  Once we're halfway
through 1st start bit, making decisions every SAMPS_BIT samples guarantees us to be
in the middle of all bits)
      UPDATE_VECTORS
      Decrement ADVANCE_COUNTER
      If ADVANCE_COUNTER == 0
            Clear ADVANCE_FLAG, set DETERMINE_FLAG
      End
```

UPDATE_VECTORS

```
      Rotate MF_COEFS around
      Shift in new sample into INPUT vector
```

COMPUTE_FILTER_OUTPUT

$z = \text{MF\_COEFS}^T$ INPUT (inner product)

DETERMINE_BIT

```
      Clear DETERMINE_FLAG
      COMPUTE_FILTER_OUTPUT
      If z > THRESHOLD
            BIT = '1'
      Else (assume z < –THRESHOLD)
            BIT = '0'
      End
      SAMP_COUNTER = SAMPS_BIT
      Decrement BIT_COUNTER
```

RESYNCHORONIZE

```
      MF_COEFS = INPUT
```

PROCESS FRAME

```
      Write ASCII code to memory (D0–D6 in text)
```