A DSP-based 16-QAM Transmitter

Dr. Thad B. Welch, PE

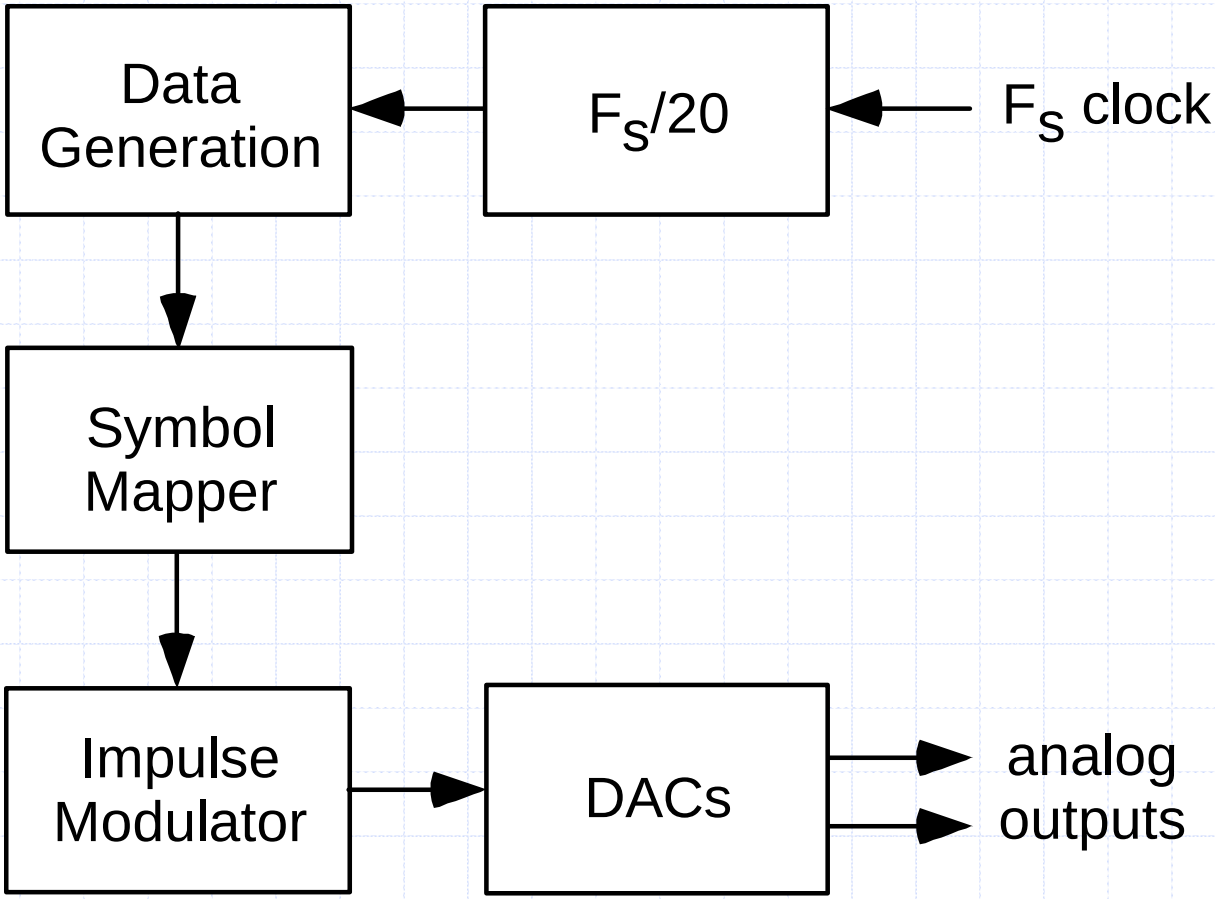
Overview

- Review of the system specifications
- System block diagram
- Implementation
- System performance
- Real-time demonstration
- Conclusions and questions

System Specifications

- System must operate at 9,600 bits/second
- Must create/generate the random data
- Signal constellation points associated with the output of the system must be as shown on page 709 of our textbook
- Scaling of the constellation is allowed

System Block Diagram



Implementation – part 1

- Data Generation
  - I added ... **#include <stdlib.h>**
    - allows for a call of **rand()**
  - **symbol = rand() % 16;**
    - modulo operations involve division
    - division is computationally expensive
  - **symbol = rand() & 15;**
    - AND with 0000 0000 0000 1111
    - results in 0, 1, 2, ..., 15
    - this is less computationally expensive

Implementation – part 2

- Symbol Mapper
  - **xLeft[0] = scaleFactor*QAM16_table[symbol];**
  - **xRight[0] = scaleFactor*QAM16_table[symbol+16];**
- QAM16_table.c (partial listing)

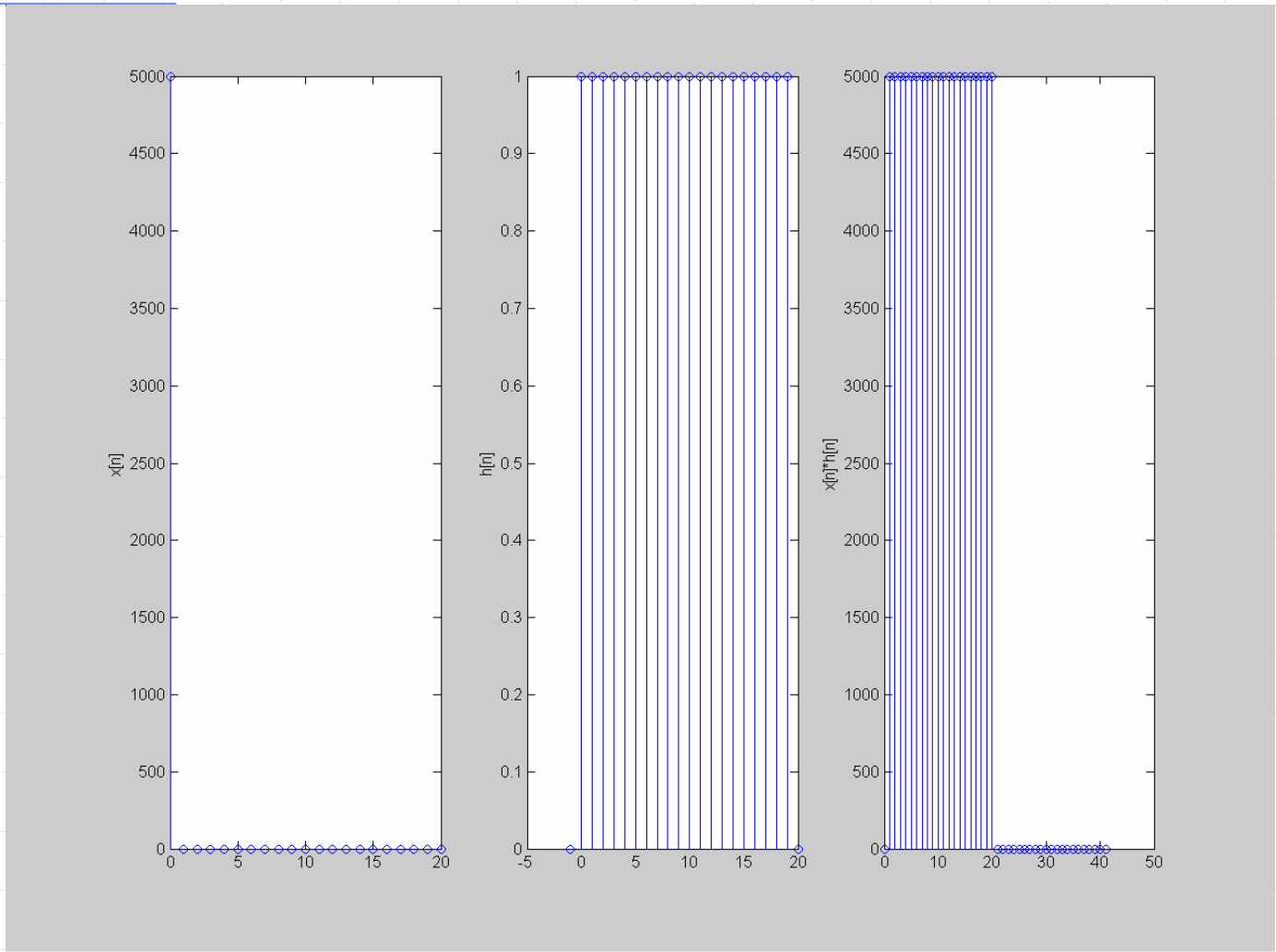
```
#include "QAM16_table.h"

float QAM16_table[32] = {
-5000,   /* QAM16_table[0]  */
-15000, /* QAM16_table[1]  */
-5000,  /* QAM16_table[2]  */ ...
```

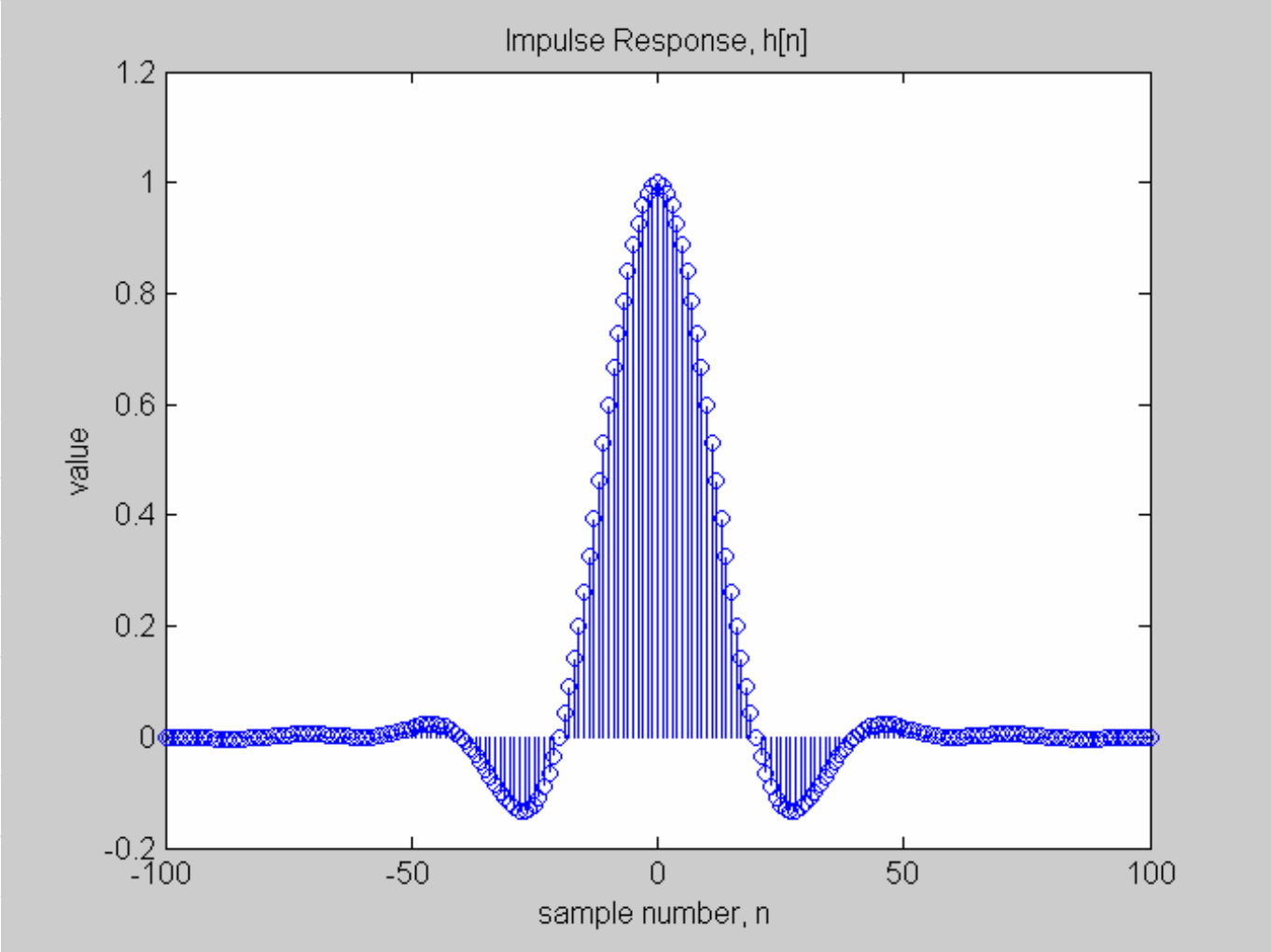
Implementation – part 3

- Impulse Modulation
  - Generate one impulse per symbol
  - The impulse's value comes from the symbol mapper
  - Use this impulse to excite an FIR filter
  - The FIR filter can have whatever shape you desire, *h[n]*
  - The output of the filter is the superposition of the individual impulse responses

Impulse Modulation Example



Raised-Cosine Pulse Shaping



Implementation – part 4

- DACs

```
else  {
  TxData.channel[LEFT]  = yLeft;

  TxData.channel[RIGHT] = yRight;
}
```

Implementation – part 5

- DACs

```
else if (outputSwitch == 2)  {
  TxData.channel[LEFT]  =
  yRight*cosTable[counter];

  TxData.channel[RIGHT] = yRight;
}
```

Implementation – part 6

- DACs

```
else if (outputSwitch == 1)  {
  TxData.channel[LEFT]  =
  yRight*cosTable[counter] -
  yLeft*sinTable[counter];

  TxData.channel[RIGHT] = yRight;
}
```

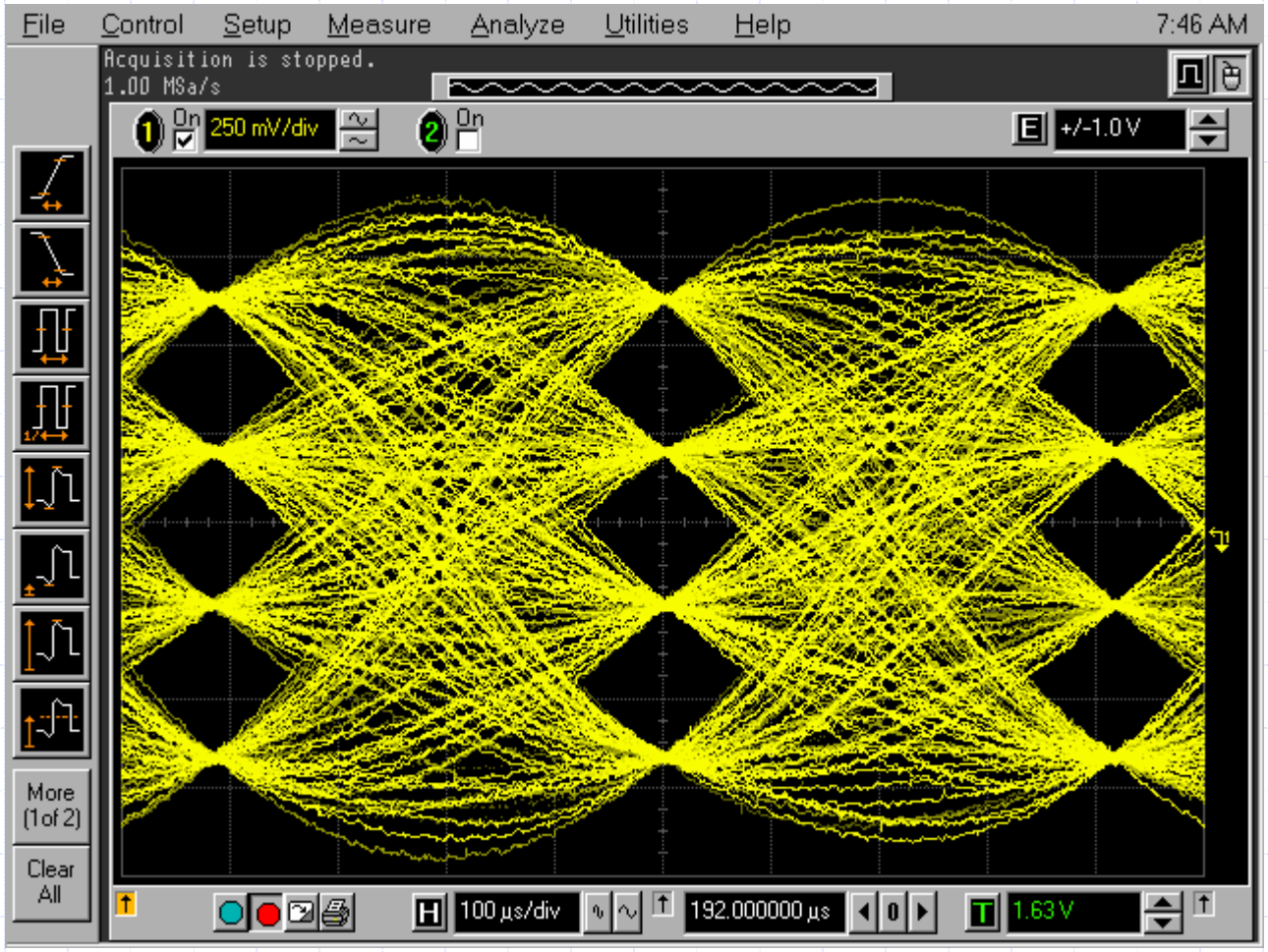
Implementation – part 7

- DACs

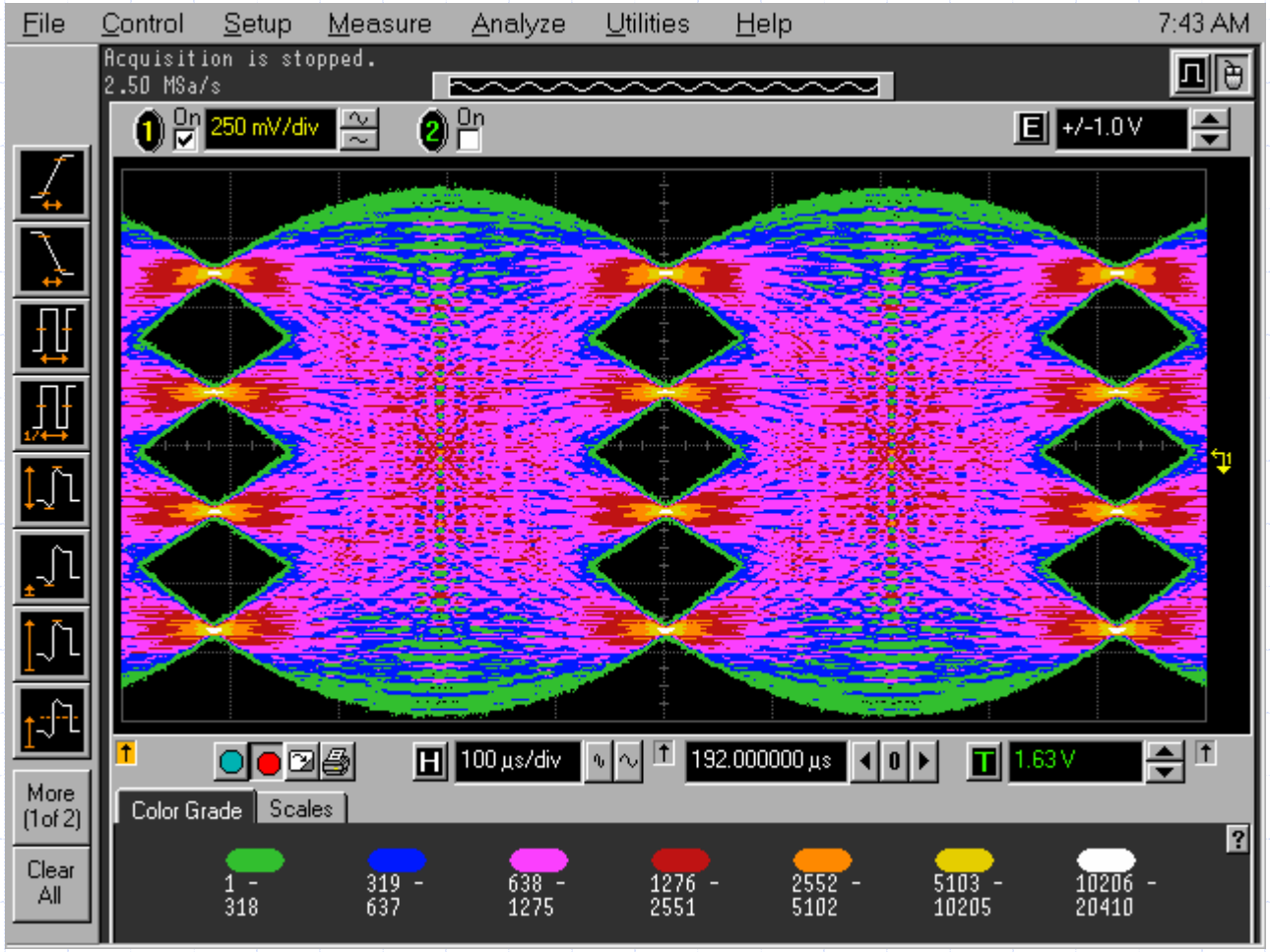
```
if (outputSwitch == 0)  {
  TxData.channel[LEFT]  =
  yRight*cosTable[counter] -
  yLeft*sinTable[counter];

  TxData.channel[RIGHT] =
  yRight*cosTable[counter];
}
```

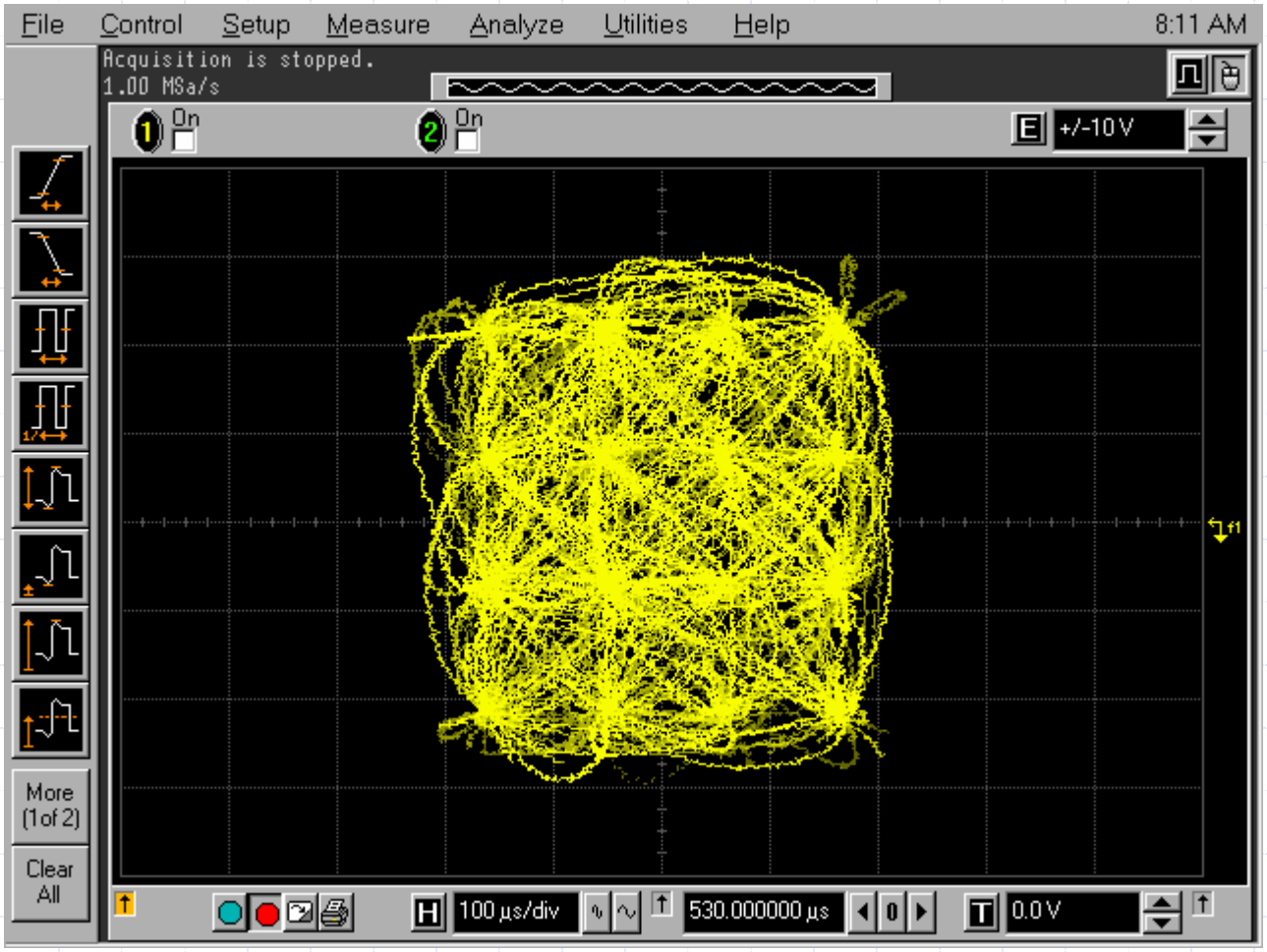
System Performance – part 1



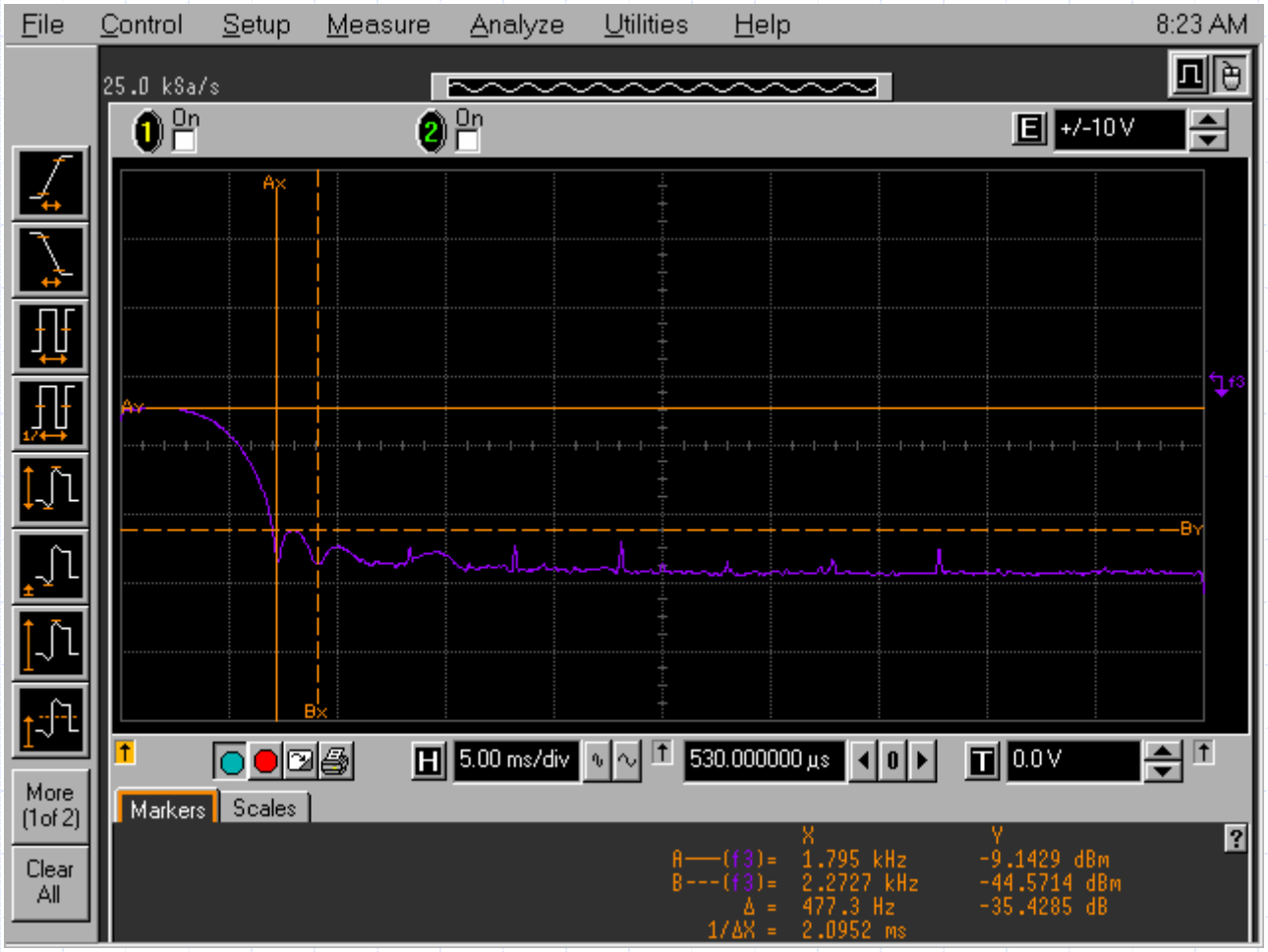
System Performance – part 2



System Performance – part 3



System Performance – part 4



Real-time demonstration

Demo Time!

Demo Time!

Demo Time!

Demo Time!

Comments

- A few comments/considerations
  - Table-based modulation/mappers
    - Once you get one modulation scheme working, others schemes can be straight-forward
  - The **volatile** keyword may be needed
    - The optimizing compiler is a mixed blessing
  - Scale factor for the system's output
  - Recalculation of filter coefficients
    - based on the roll-off factor, *r*
  - Impulse modulation isn't that expensive

Comments and Conclusions

- Modulation – $\sin\left(n\frac{\pi}{2}\right)$ and $\cos\left(n\frac{\pi}{2}\right)$
- CODEC pin for symbol synchronizing pulses
- Serial port issues
  - ISRs must be allowed to complete
- Output switch
  - Selectable outputs without a code rebuild
  - Windows-based Apps are now possible

DSP-based 16-QAM Transmitter

Questions?