LABORATORIUM DIGITALE SYNTHESE

ONTWERP VAN EEN “WIRELESS SPREAD SPECTRUM” COMMUNICATIE SYSTEEM.

INLEIDING :

Het laatste decennium kende de draadloze communicatie een explosieve groei. Ook de symbiose tussen de “**wireless communications**” en de digitale micro elektronica evolueert zeer snel. De diverse mobile communicatie systemen vormen nu samen met de computer industrie de drijfveer voor de ontwikkeling van “**high-speed**” en “**low-power**” geïntegreerde schakelingen. Recent zijn door deze ontwikkelingen toepassingen met “**Spread-Spectrum**” (SS) systemen economisch mogelijk geworden. De techniek van **SS** laat diverse communicatie toepassingen toe, gaande van “**satelite-based spread-spectrum systems**” tot “**high-capacity wireless office telephone systems**”. **SS** kan gedefinieerd worden als een techniek waarbij een **hulp modulatie golfvorm** onafhankelijk van de informatie data wordt gebruikt voor het **spreiden** (spread) van de **signaal energie** over een **veel grotere bandbreedte** dan de oorspronkelijke informatie data bandbreedte. Aan **zender** zijde zal zo een **spread** operatie worden uitgevoerd, terwijl aan **ontvanger** zijde een **despread** operatie zal uitgevoerd worden. De uitdaging van de ontvanger bestaat er echter in om een **gesynchroniseerde replica** van de **hulp modulatie golfvorm** te genereren. Spread-spectrum bezit een aantal **eigenschappen** waardoor de communicatie techniek bruikbaar wordt voor :

- *Verbergen van het signaal (onder de ruisvloer).*
- *Geen interferentie met conventionele systemen.*
- *Ant-jam en interferentie onderdrukking.*
- *Multiple access.*
- *Multipath mitigation.*

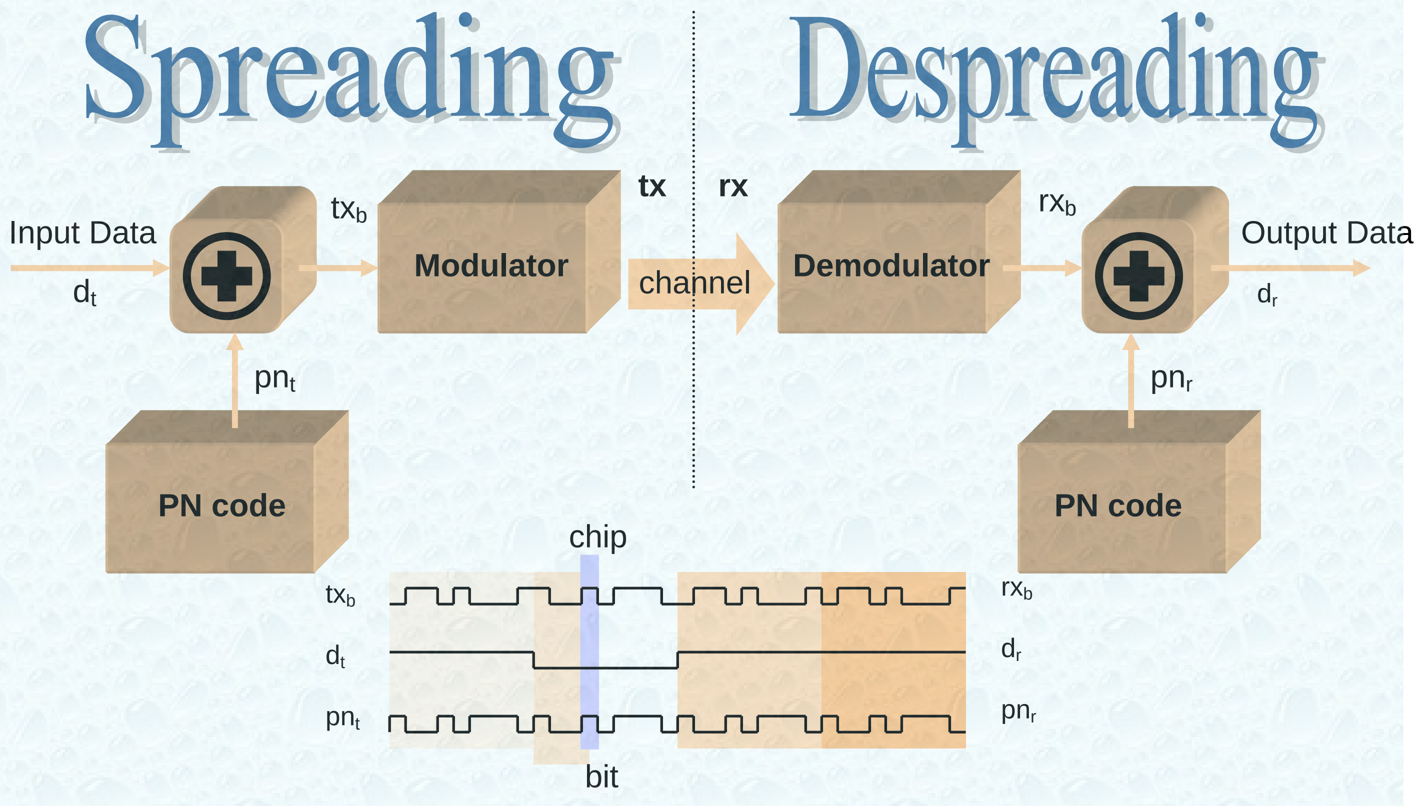


Fig. 1 : Direct Sequence Spread Sprectrum

Er zijn twee verschillende **SS types** : "**frequency hopping**" (FH) en "**direct sequence**" (DS).

Bij **DS spread spectrum** wordt het data signaal in fase gemoduleerd met een "**pseudo-noise**" (**PN**) ook wel "**pseudo-random**" sequentie van nullen en eenen. Deze worden **chips** genoemd. The chip modulatie is meestal **BFSK**. De PN chip sequentie wordt **mod-2** opgeteld met de datastroom. Het aantal PN chips per bit bepaald de "**processing gain**". DS systemen kunnen in twee categorieën worden onderverdeeld : "long en short code systemen". "**Long code**" systemen hebben een code lengte welke veel groter is dan deze van een data symbool. Hierdoor wordt een verschillend chippatroon met elk symbool geassocieerd. "**Short code**" systemen gebruiken repeterend hetzelfde chippatroon voor een data symbool. "Short code" systemen maken meestal gebruik van "**matched-filter**" detectie. "Long code" systemen daarentegen gebruiken meestal een "**correlation**" detectie. "Short code" systemen zijn instaat een **snelle acquisitie** te verwezenlijken en zijn daarom geschikt voor "**burst traffic**" zoals in besturings- en controlesystemen.

Bij **FH spread spectrum** wordt de beschikbare bandbreedte verdeeld in N kanalen. In functie van de PN code welke bekend is door modulator (zender) en demodulator (ontvanger) "**hopped**" men tussen deze N kanalen. Omdat FH meestal resulteert in een fase discontinuïteit zal dikwijls een niet coherente modulatievorm gebruikt worden zoals **MFSK** of **DPSK**. FH systemen kunnen worden opgesplitst in "slow- en fast hopping" (relatief t.o.v. de datasymbool rate). Bij "**slow hopping**" zijn er meerdere datasymbolen per hop, terwijl bij "**fast hopping**" zijn er meerdere hops per datasymbool. MFSK zal meestal gebruikt worden bij "fast hopping" en DPSK gewoonlijk bij "slow hopping".

DOEL :

- ✓ Het leren omgaan met de **hardware ontwerp cyclus**.
- ✓ Het leren opstellen van een **systeem concept**.
- ✓ Het leren omgaan met de hardware beschrijvingstaal **VHDL** als hedendaagse **digitale ontwerp-** en **synthese techniek**.
- ✓ Het leren **simuleren** van de hardware beschrijving aan de hand van het oordeelkundig opstellen van een **set testvectoren**.
- ✓ Inzicht verwerven in een subset datacommunicatie begrippen om een efficiënt hardware ontwerp mogelijk te maken.

  V.b. : *Direct Sequence Spread Spectrum, application layer, datalink layer, acces layer, physical layer, OOK, PN code, m-sequence en Gold code, matched filter, NCO, DPLL, sampling, auto- en cross correlatie, integrate and dump, synchronisatie, PRBS, BERT, error performantie, NRZ en RZ, jitter analyse…*
- ✓ Implementatie van digitale systemen d.m.v. "**F**ield **P**rogrammable **G**ate **A**rray's" **FPGA** devices.
- ✓ Het leren uitvoeren van **functionele testen** op de gerealiseerde circuits.
- ✓ Het leren uitvoeren van een **systeemtest** door met de volledige systeem specificaties rekening te houden.
- ✓ Het leren gebruiken van een **L**ogic **A**nalyser en **P**attern **G**enerator bij het testen van het systeem.
- ✓ Het leren gebruiken van een **D**igital **T**ransmission **A**nalyzer bij het meten van de error performantie van het communicatie systeem.

VOOR TE BEREIDEN :

- ✓ Lees de tekst "VHDL" van D. Van Landeghem en het gelijknamig hoofdstuk uit de cursus "Digitale Synthese" van J. Meel.

- ✓ Lees de tekst “Finite State Machine” van J. Meel.
- ✓ Bestudeer de tekst “Spread Spectrum (SS) introduction” van J.Meel, D. Van Landeghem en A. Breckpot.
- ✓ Zoek zelf bijkomende informatie betreffende SS in de vakliteratuur.
- ✓ Bestudeer de FPGA architectuur in bijgaande “datasheets”.
- ✓ Bestudeer in bijlage het systeem concept van de “Elementary Spread Spectrum demonstrator” van D. Van Landeghem.
- ✓ Lees “VHDL Design Flows” door Dave Van den Bout (XESS Corporation).
- ✓ Lees de documentatie van de XS40 designkit.

OPDRACHT :

- ✓ Systeem specificaties Baseband **TX** :
    - De zender kan in de application layer een **4 bit commando** genereren overeenkomstig elk van de 4 schakelaars (zie fig. “Spread Spectrum TX : Baseband”).
    - Zolang een schakelaar actief is zal het overeenkomstig commando (4 bit symbool) repeterend worden aangeboden aan de RF transmitter (zie fig. “Spread Spectrum Wireless Communication”).
    - De **PN generator** is een **4 bit m-sequence** systeem dat bijgevolg een **patroon** van **15 chips** (PN-sequence) omvat. M.a.w. **Tchip = Tbit / 15** en **Tbit = Tcmd / 4**. Bepaal aan de hand van deze gegevens en de fig. “Spread Spectrum Wireless Communication” de freq. van de **systeem clk**.

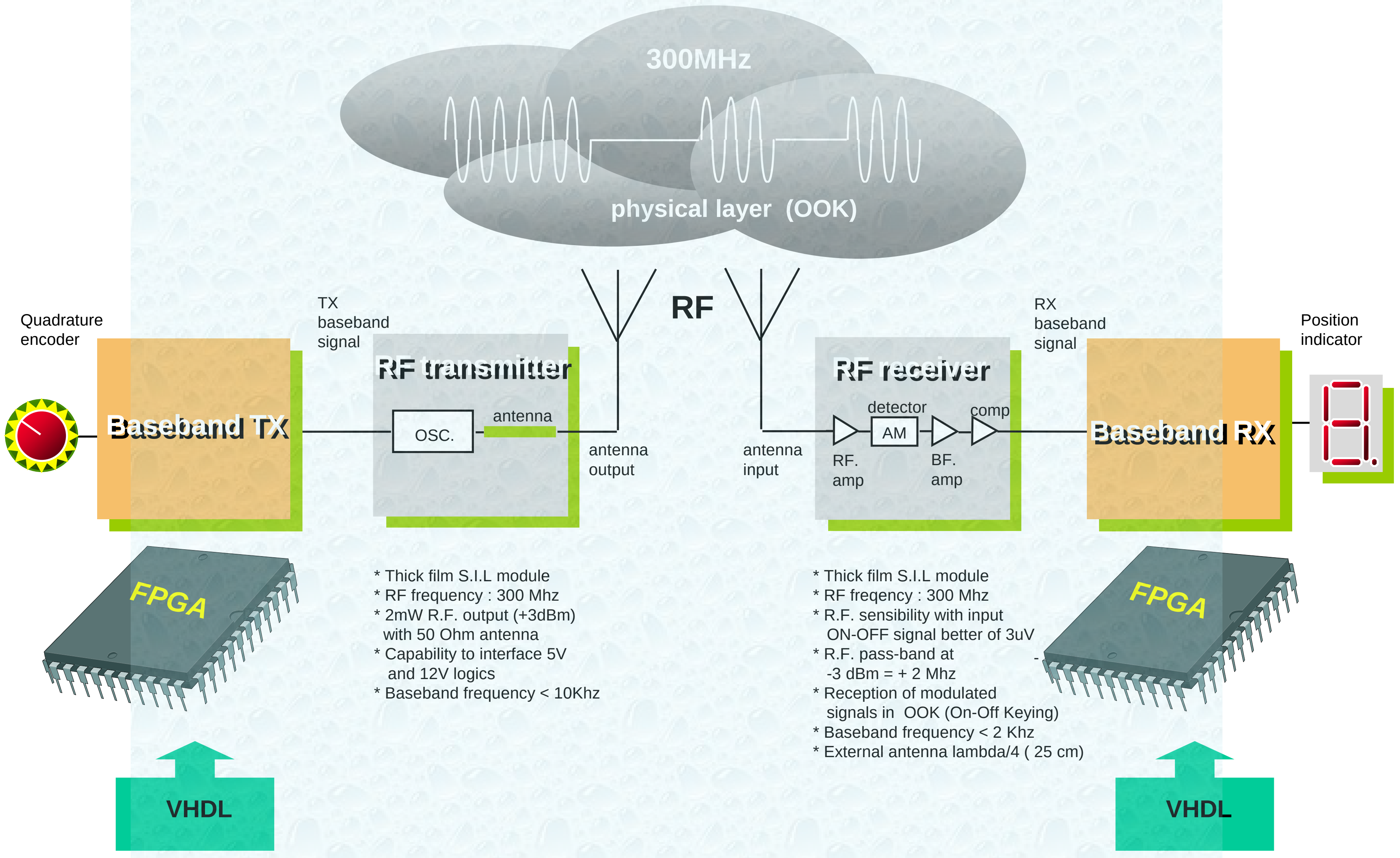
- ✓ Systeem specificaties Baseband **RX** :
    - In de application layer zal bij ontvangst van een juist commando de overeenkomstige commando LED branden (zie fig. . “Spread Spectrum RX : Baseband”).
    - De ontvanger beschikt in de acces layer over een **DPLL** (Digitale Phase Locked Loop). Deze draagt zorg voor de **synchronisatie** van de **PN-sequence replica** met de zender. De DPLL kan “worst case” binnen **3 chip transities** synchroniseren.
    - De DPLL beschikt hiervoor over een **NCO** (Numerical Controlled Oscillator) welk als een preloadable counter kan worden geïmplementeer. De preload waarde wordt bepaald door een circuit welke het verschil meet tussen het tijdstip van de chip sample en dit van de chip transitie (zie fig. “Baseband RX Chip Synchronisatie”). Tijdens zowel de acquisitie- als de tracking fase zal de synchronisatie op 5 LED’s worden uitgelezen overeenkomstig de 5 gedefinieerde chip segmenten.
    - Voor de acquisitie wordt een **matched filter** gebruikt.
    - De despread data wordt aan de datalink layer gegeven (zie fig. “Baseband RX Chip Synchronisatie”) via een **correlator** van het type “**integrate and dump**” (zie fig. “Baseband RX Correlator (integrate and dump)”). Deze correlator kan geïmplementeerd worden als een up/down counter.

- ✓ Systeem opbouw :
    - Maak voor de opgegeven systeem specificaties van de baseband TX en baseband RX een **systeem concept**. Aangezien de omvang en de complexiteit van de opdracht mag ook het systeem concept in bijlage worden gebruikt. (Eigen initiatieven zijn daarentegen altijd welkom desnoods voor delen van het ontwerp.)
    - Beschrijf de hardware van de baseband TX en van de baseband RX elk apart in een **VHDL** file. *Ga hierbij systematisch tewerk en beschrijf b.v. eerst een eenvoudig*

functioneel blok van de baseband TX. Verifieer (simuleer) deze eenvoudige functie en breidt daarna de baseband TX beschrijving stelselmatig uit tot de volledige baseband TX is beschreven met VHDL.
Ga op gelijkaardige manier tewerk bij de beschrijving van de baseband RX.

- Stel een gedetailleerde **set testvectoren** op voor zowel baseband TX als baseband RX en **simuleer** beide schakelingen.
- **Synthetiseer** de hardware voor de TX en voor de RX en **implementeer** deze elk in **XILINX FPGA** van het type **XC4010E**.
- **Download** de **bitmap** files in de design kit voor de TX en een voor de RX.
- **Debug** de TX en de RX gebruikmakend van de **logic analyser en pattern generator**. Voer daarna een **systeemtest** uit.
- Controleer daarna met de **Transmission Analyser** de communicatie performantie. Leg hierbij aan de baseband TX een **PRBS** (pseudo random bit sequence) data signaal en PRBS clk signaal aan volgens fig. Measuring communication performance. Het meetprincipe hierbij is in fig. BER testing terug te vinden. Welke **BER** (bit error rate) waarde vindt u na 1 kwartier? Is deze waarde realistisch?
- Gradaties in de error metingen zijn in fig. Error performance terug te vinden. Gebruik als **jammer** de Signal Vector Analyser voorzien van twee logperiodieke antennes, een voor ontvangst en een voor te zenden (jammer).

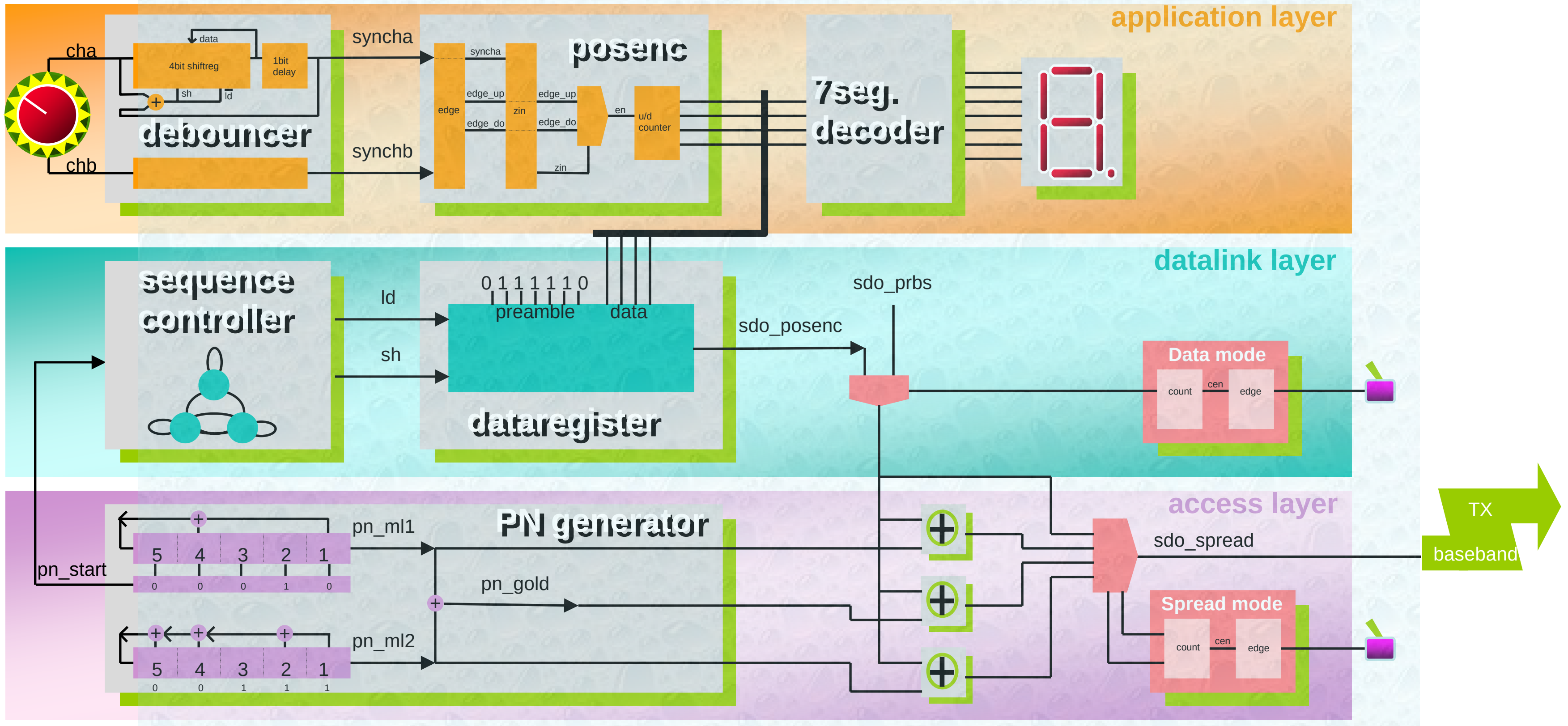
Spread Spectrum Wireless Communication

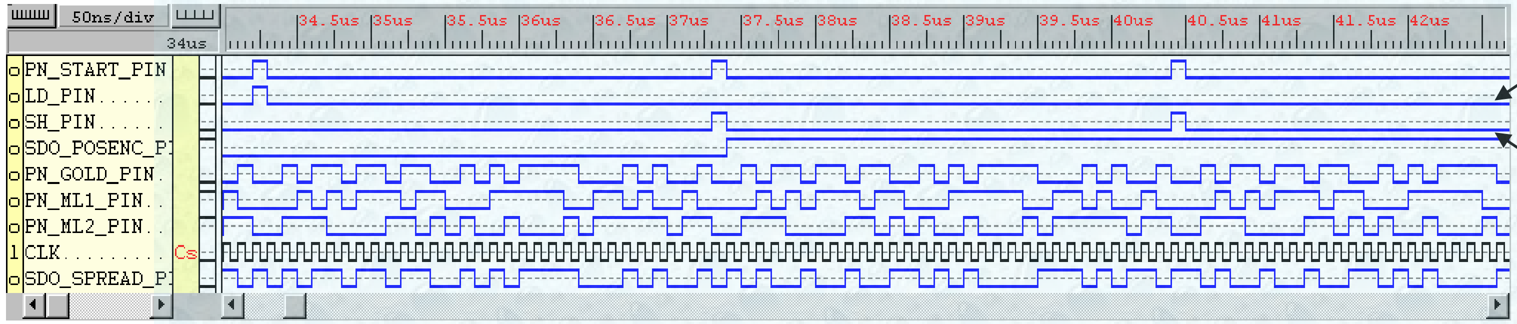


* Thick film S.I.L module
* RF frequency : 300 Mhz
* 2mW R.F. output (+3dBm)
  with 50 Ohm antenna
* Capability to interface 5V
  and 12V logics
* Baseband frequency < 10Khz

* Thick film S.I.L module
* RF freqency : 300 Mhz
* R.F. sensibility with input
  ON-OFF signal better of 3uV
* R.F. pass-band at
  -3 dBm = + 2 Mhz
* Reception of modulated
  signals in OOK (On-Off Keying)
* Baseband frequency < 2 Khz
* External antenna lambda/4 (25 cm)

Spread Spectrum TX : Baseband

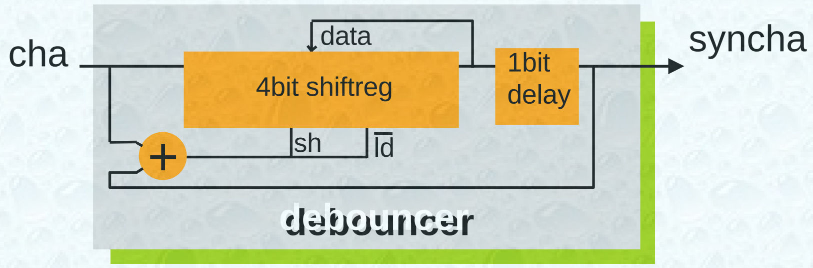


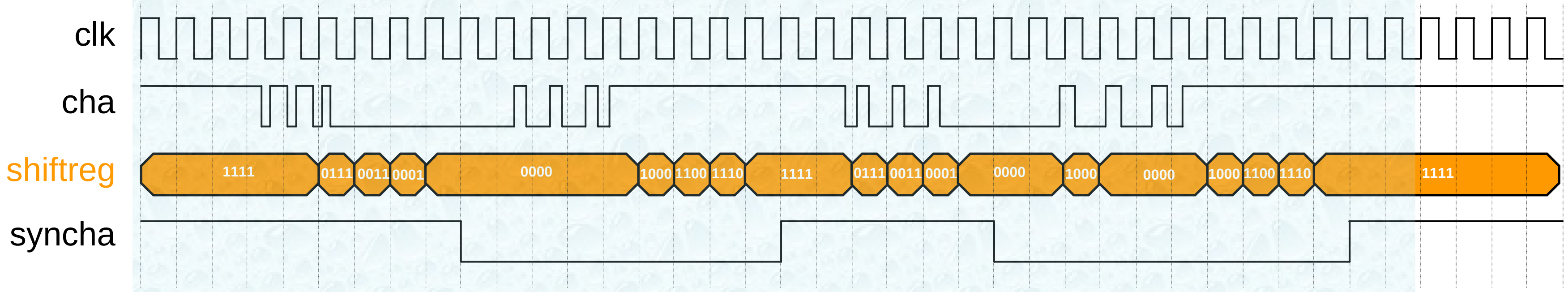


Frame sync.

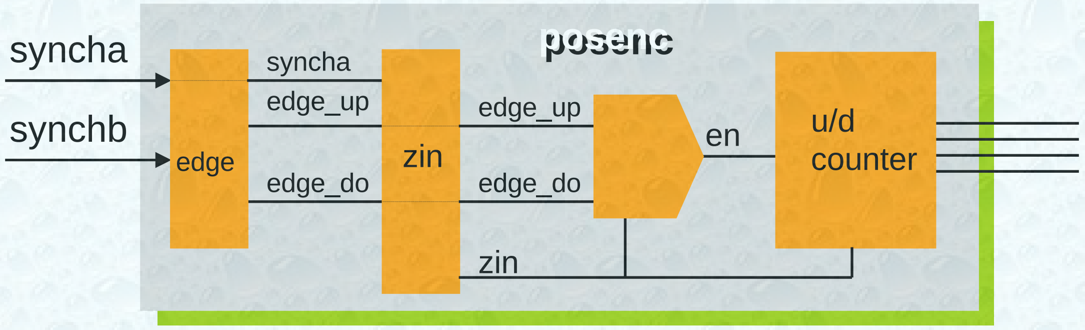
Bit sync.

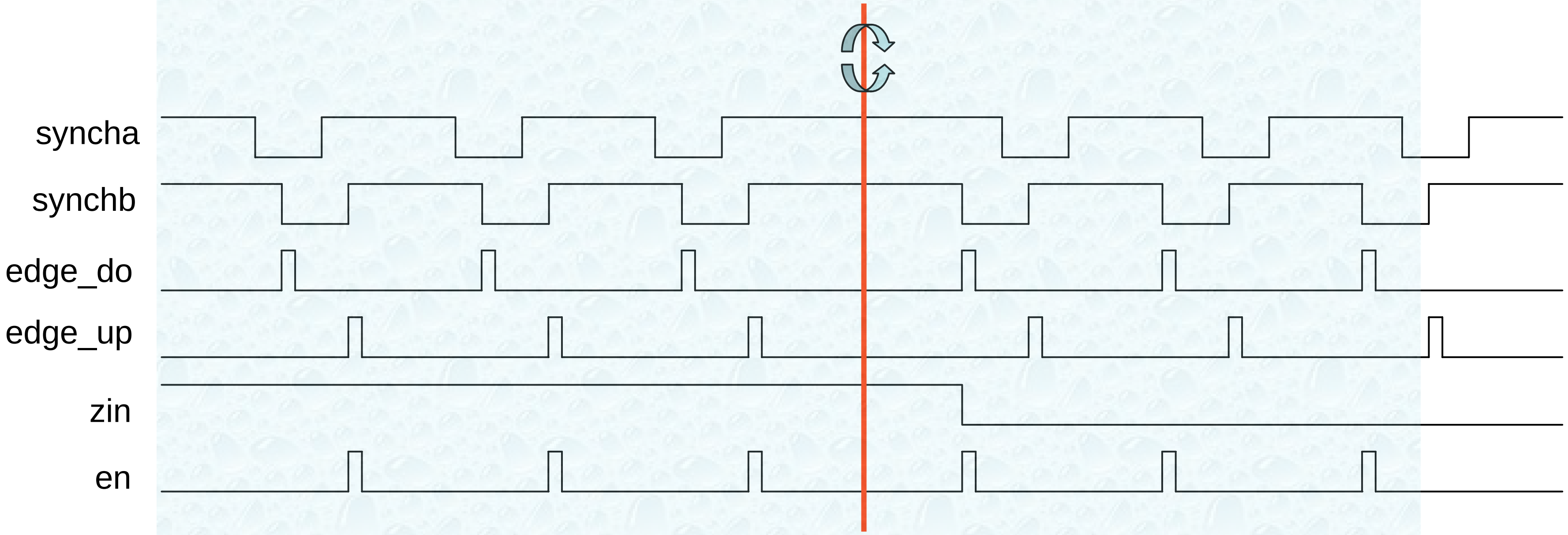
Spread Spectrum TX : Debouncer



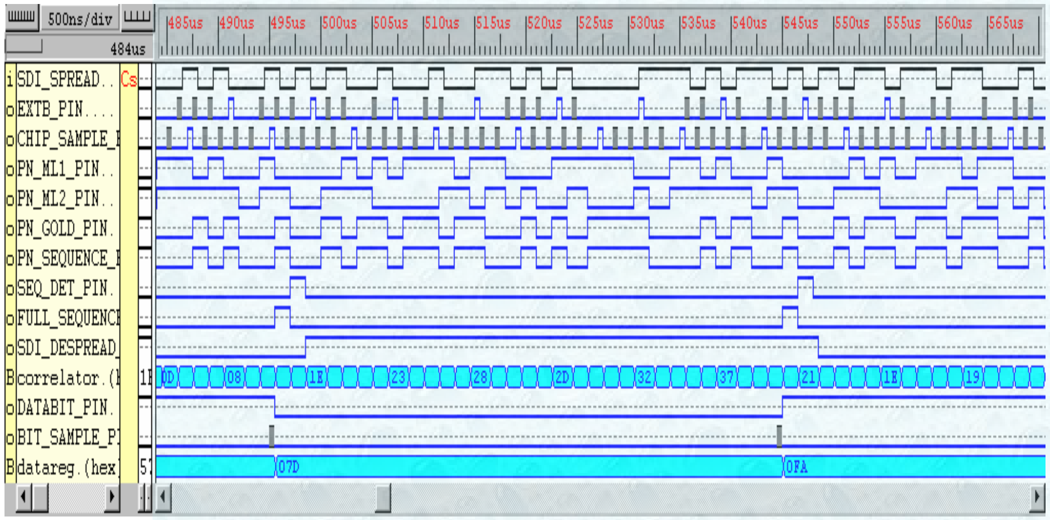


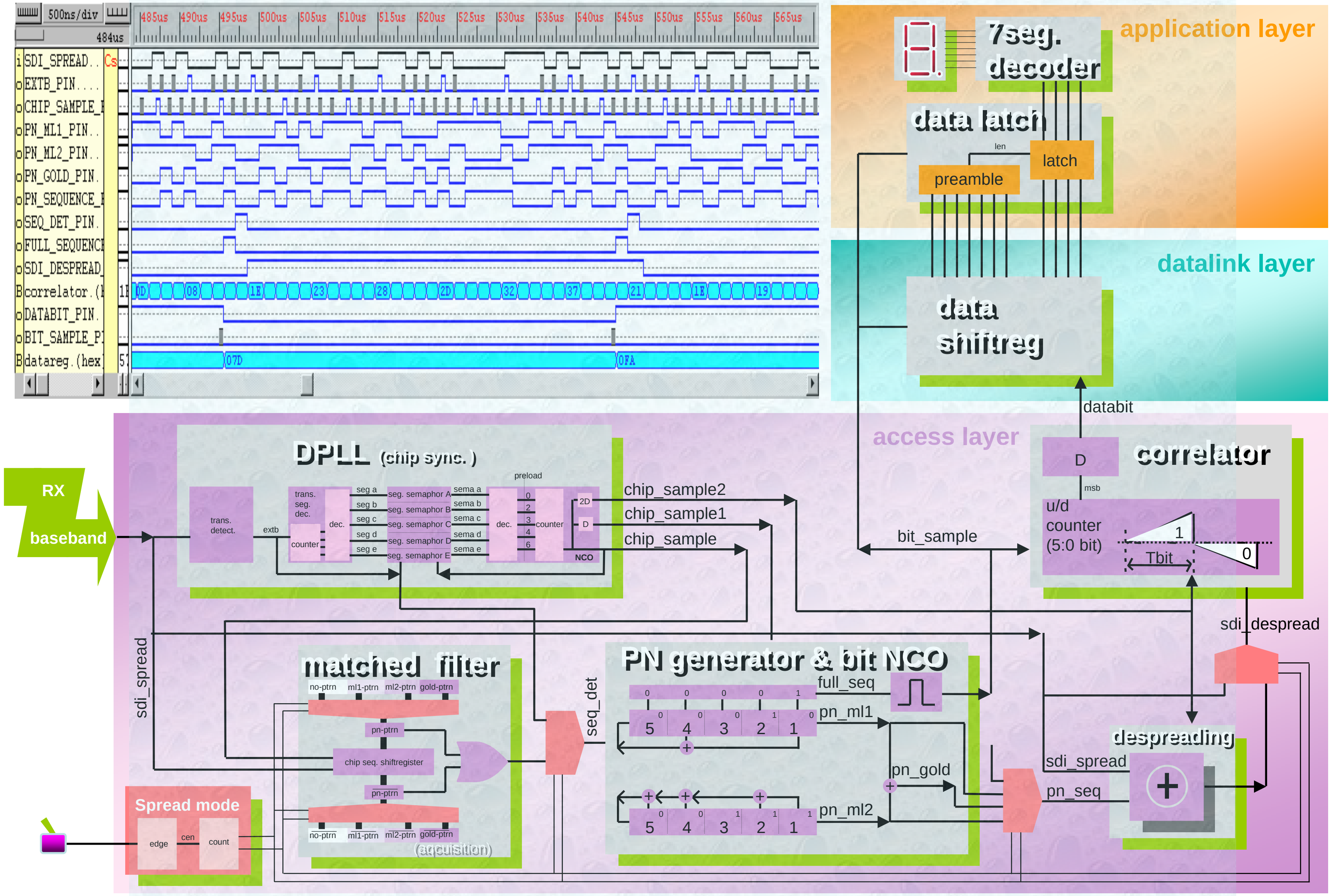
Spread Spectrum TX : Position encoder



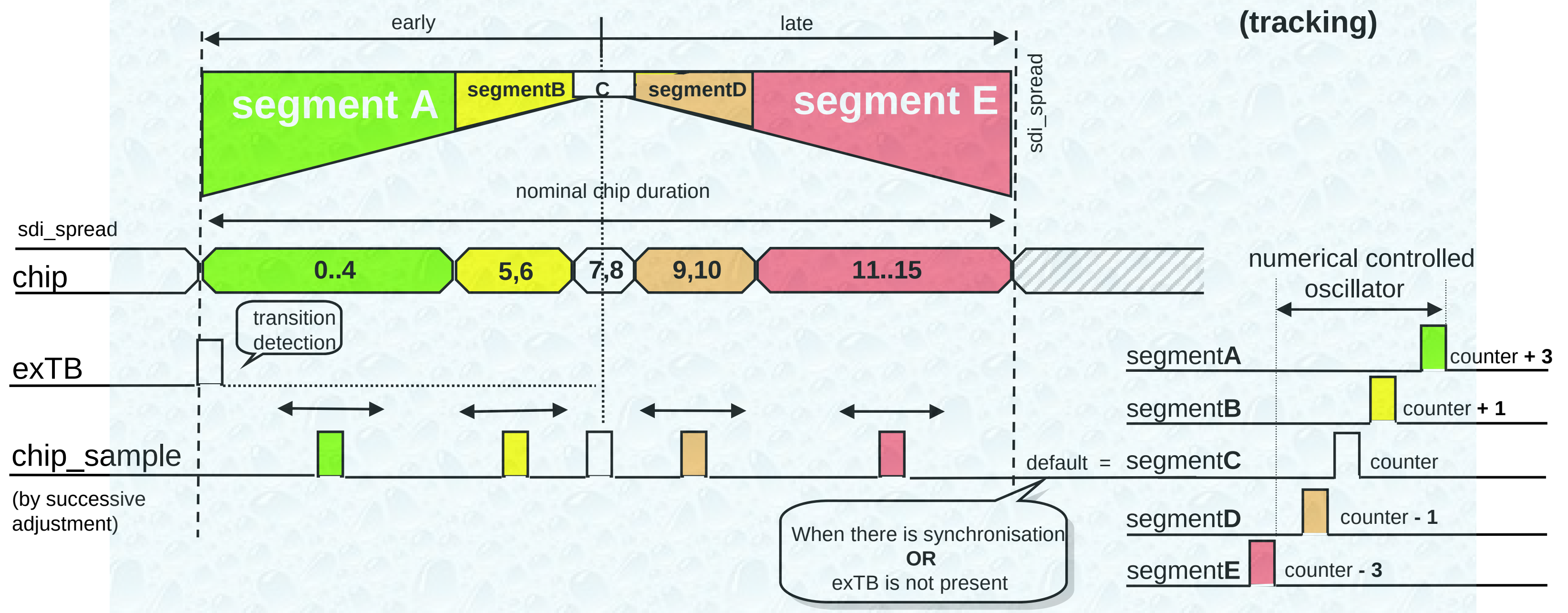


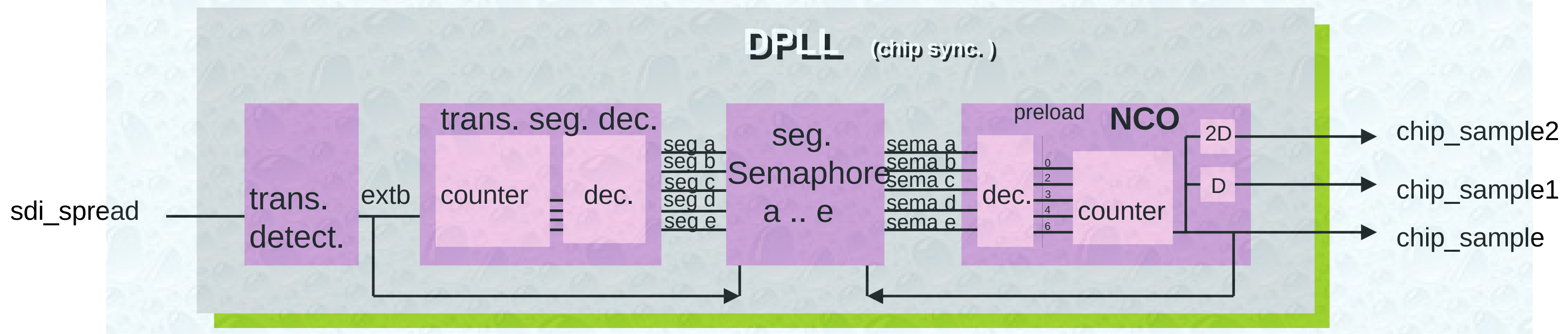
Spread Spectrum RX : Baseband



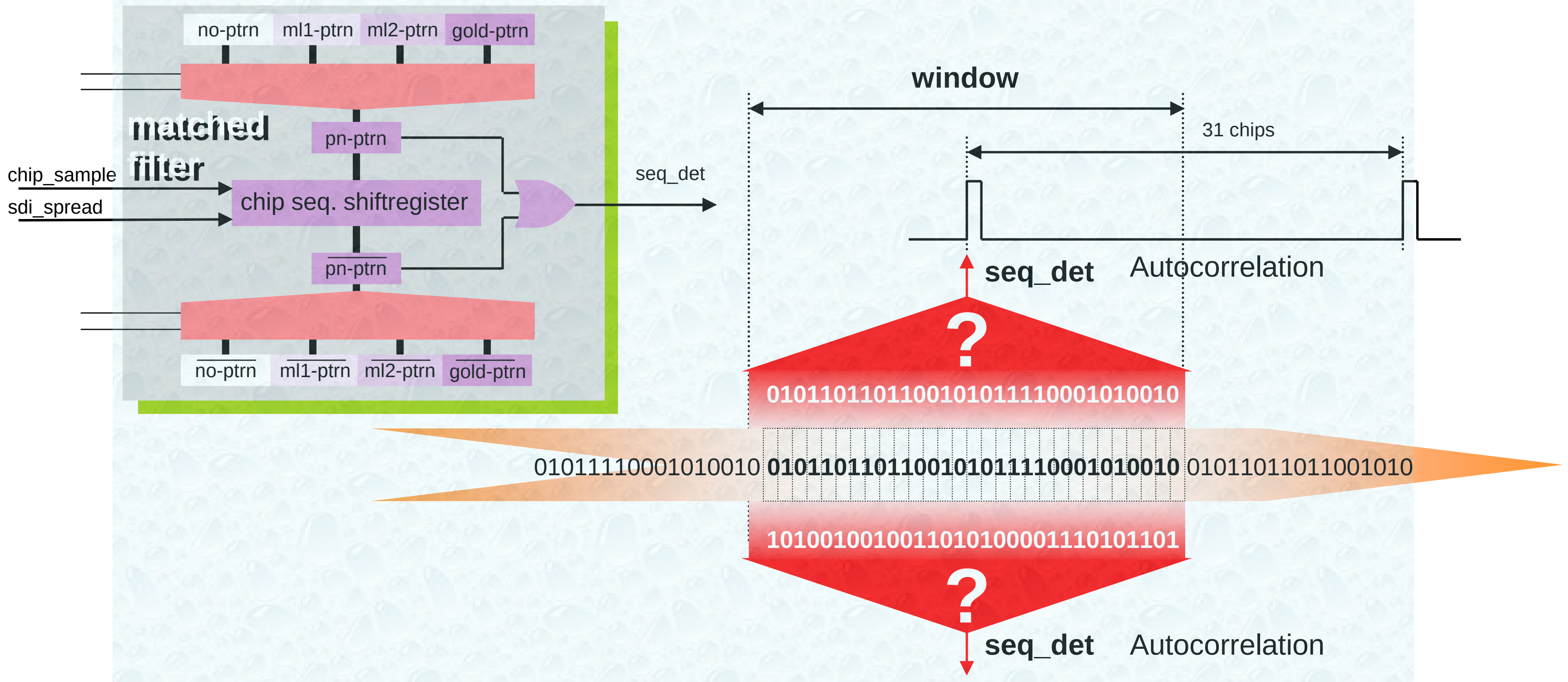


Baseband RX Chip Synchronisation

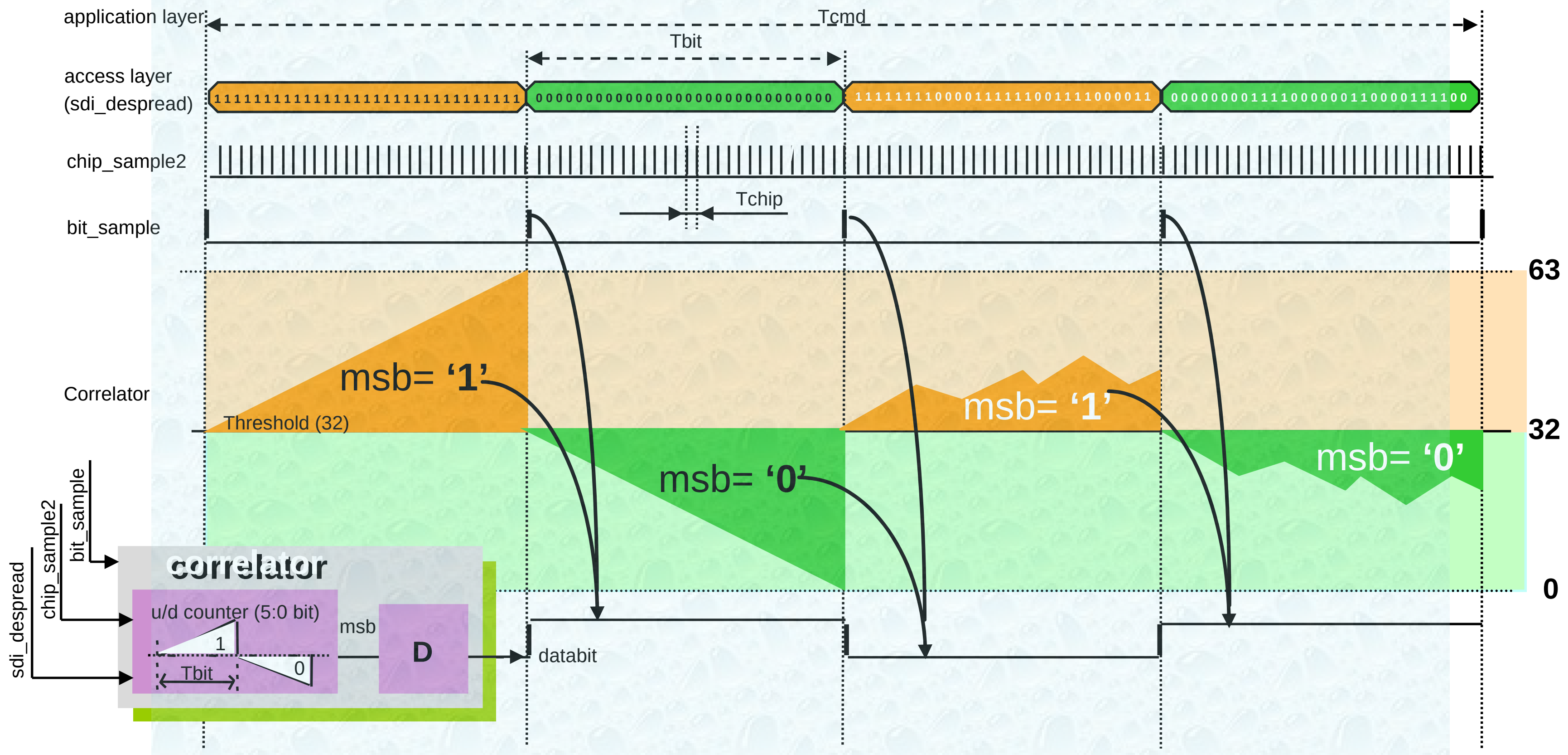




Baseband RX PN matched filter (acquisition)

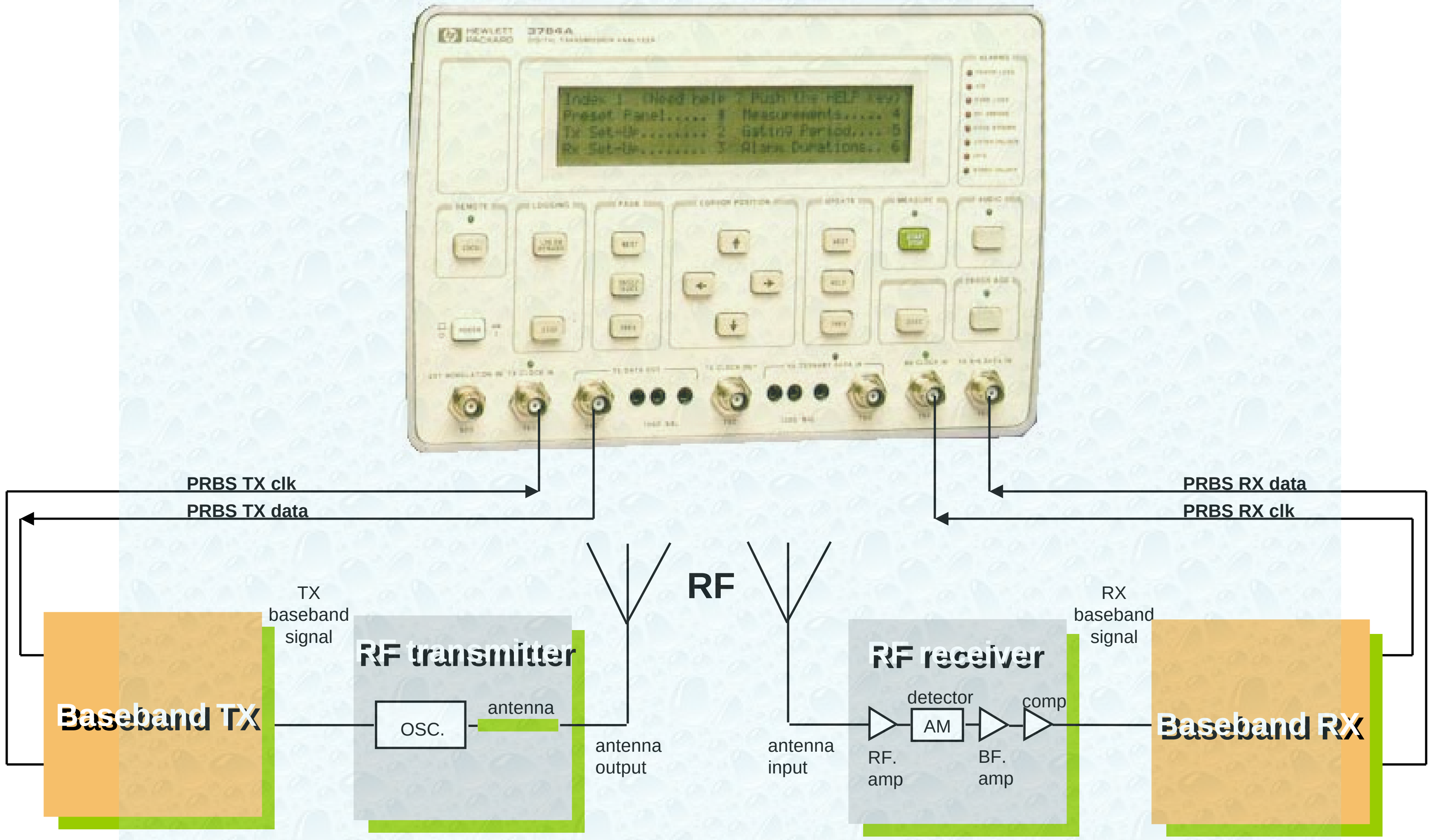


Baseband RX Correlator (integrate & dump)



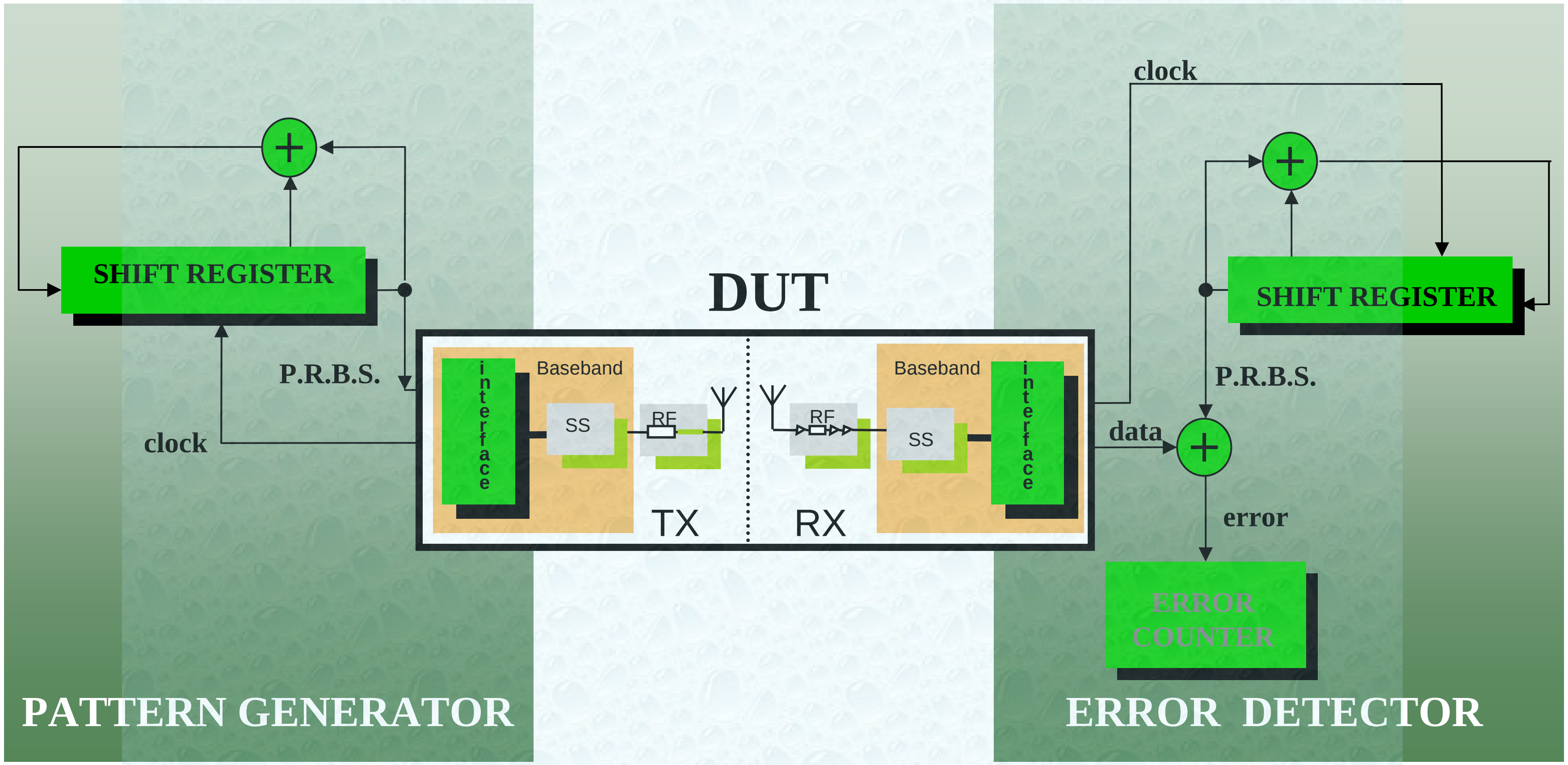
Wireless Spread Spectrum

Measuring communication performance



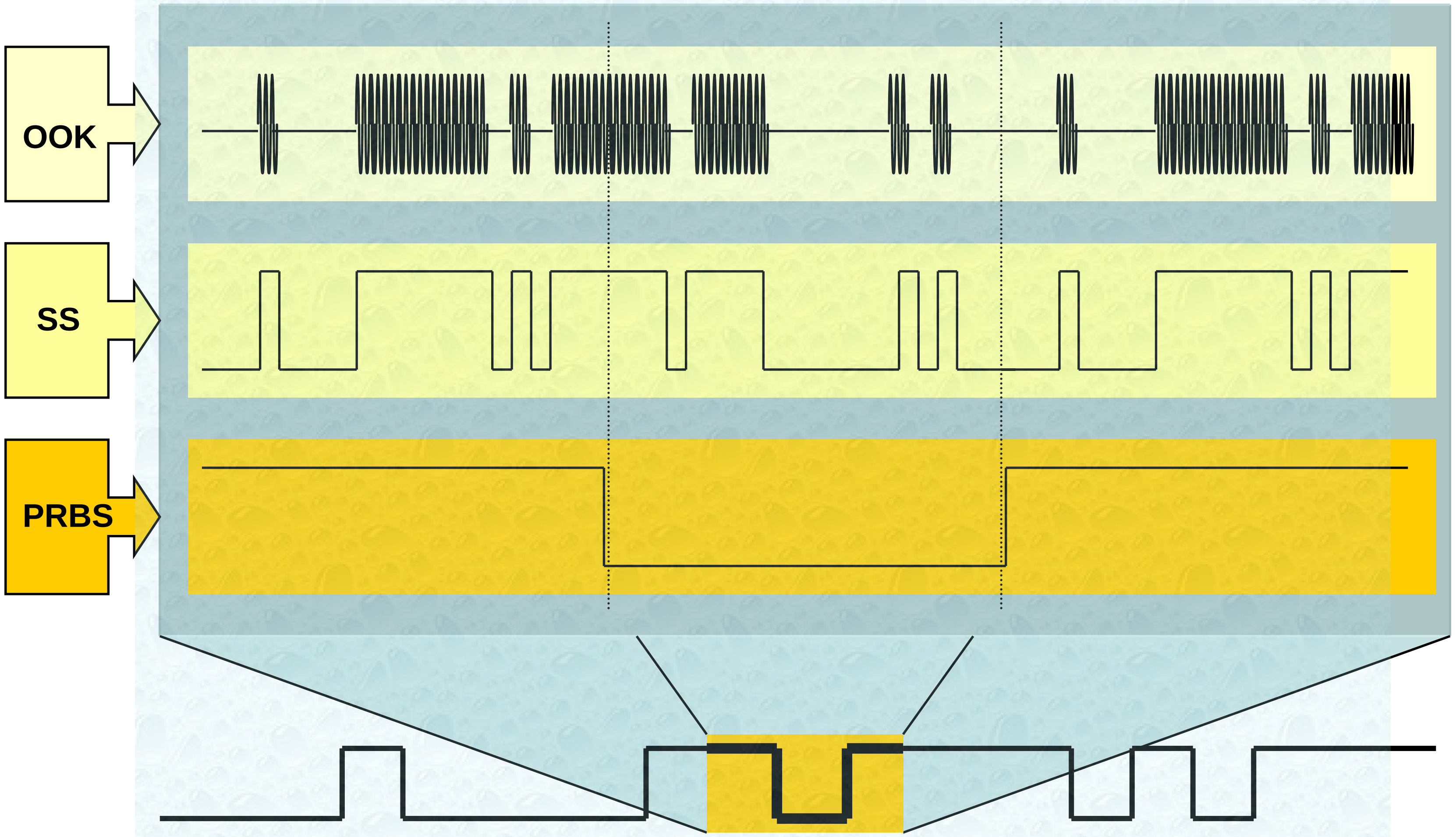
Wireless Spread Spectrum

BER testing



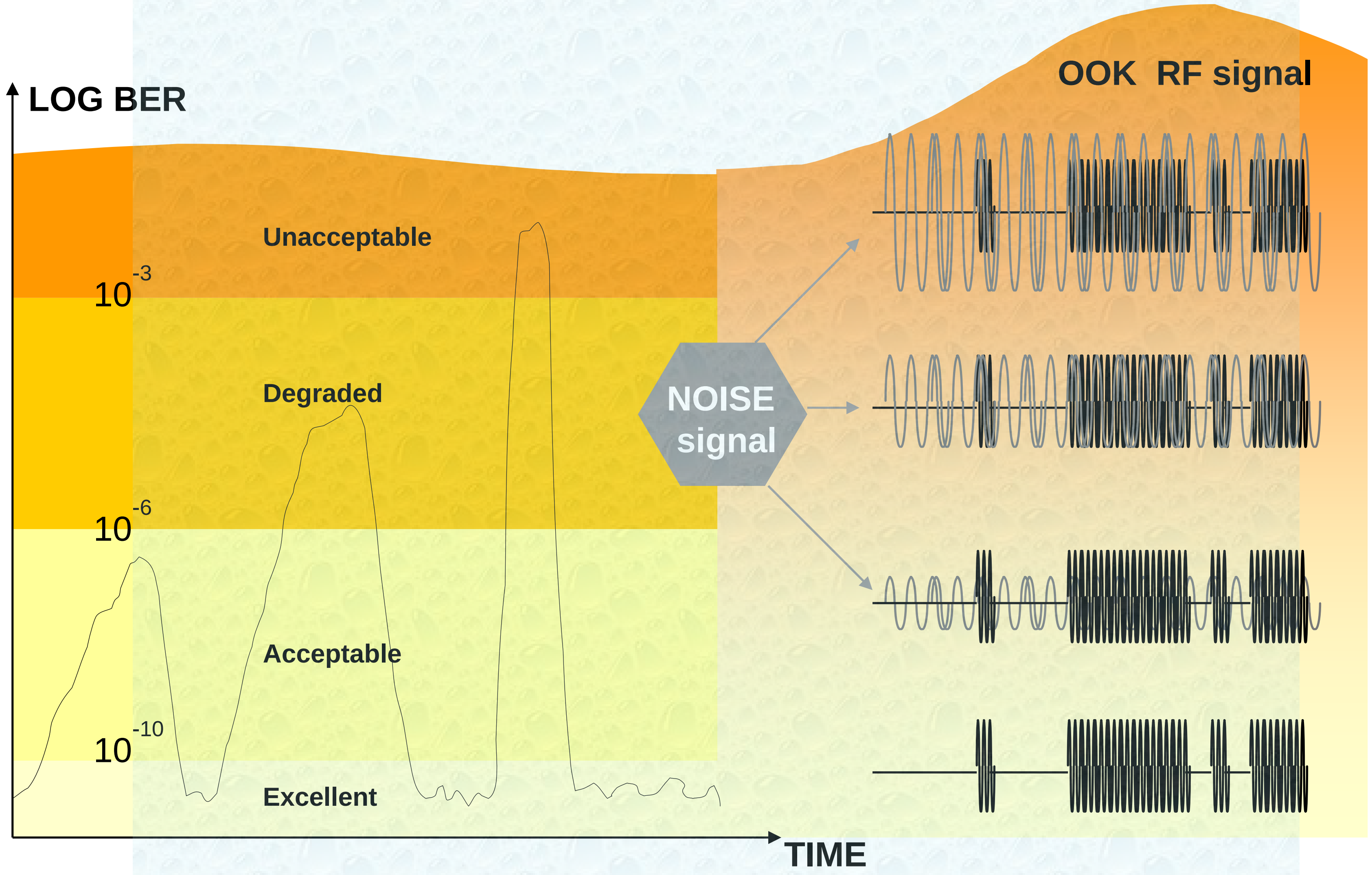
Wireless Spread Spectrum

Pattern Overview



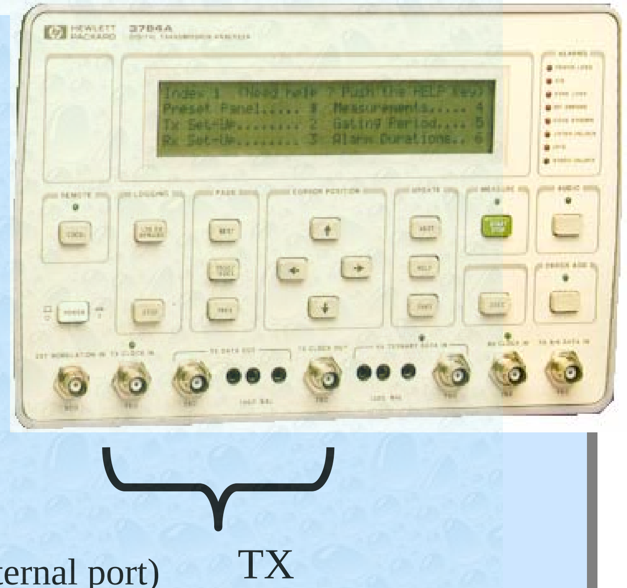
Wireless Spread Spectrum

Error performance



Wireless Spread Spectrum

Digital Transmission Analyzer (error measurement)



Transmitter

Clock : -Internal Fixed Rates
(704, 2048, 8448, 34368 kb/s)
- Internal Variable Clock
(1 kb/s to 50 Mb/s)
- External Clock(1 kb/s to 50 Mb/s)

Test Patterns : - PRBS (2^n - 1, where n = 6, 9, 11, 15, 17, 20, 23)
(0 to 999 zero substitution)
- Word (1 to 16 bit fully programmable)
(2 programmable 8-bit words alternated via an external port)

Error injection : - Types : bit or code
- Add : single or at fixed rate of 1 in 10^n, n = 3, 4, 5, 6

Outputs : - clock : level is TTL or ECL
- data : binary mode --> NRZ : level is TTL or ECL
terary mode --> RZ : line code is AMI or HDB3

Digital Transmission Analyzer (error measurement)

Receiver

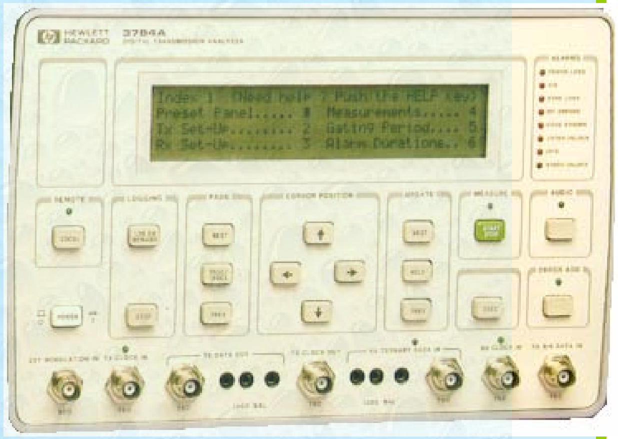
Inputs :
- clock : level is TTL or ECL, frequency range 1kb/s to 50Mb/s, polarity is norm. or inv.
- data : binary mode --> NRZ : level is TTL or ECL
  terary mode --> RZ : line code is AMI or HDB3

Measurements :
- bit and code error count, -ratio, -intervals (seconds or deciseconds), -error free intervals
- slip deciseconds
- duration of power loss, sync. loss, AIS,
  data loss and clock loss
- received clock rate and offset measurement

Error Performance Analysis :
- %unavailability, %errored seconds …
- long-term mean error ratio

Real-time Clock :
  Day, month, year, hours, minutes and seconds.

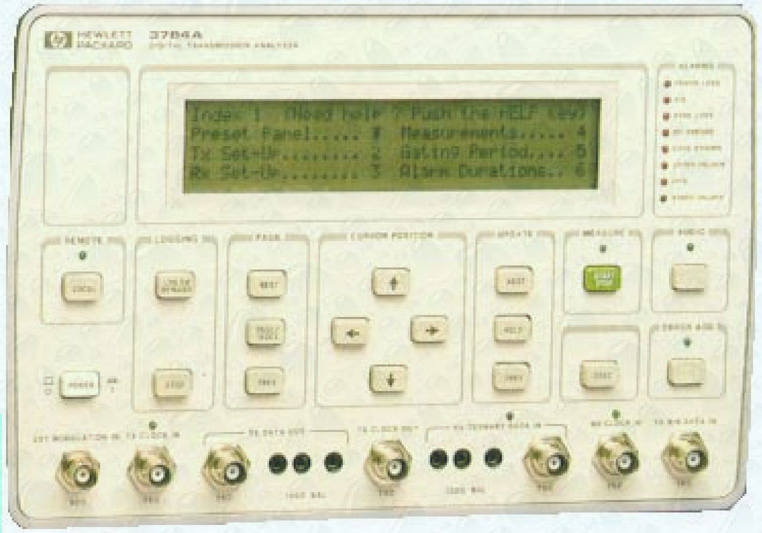
Gating :
  Manual, single or repeating (1 second to 100 days)



RX

Wireless Spread Spectrum

Digital Transmission Analyzer (jitter measurement)



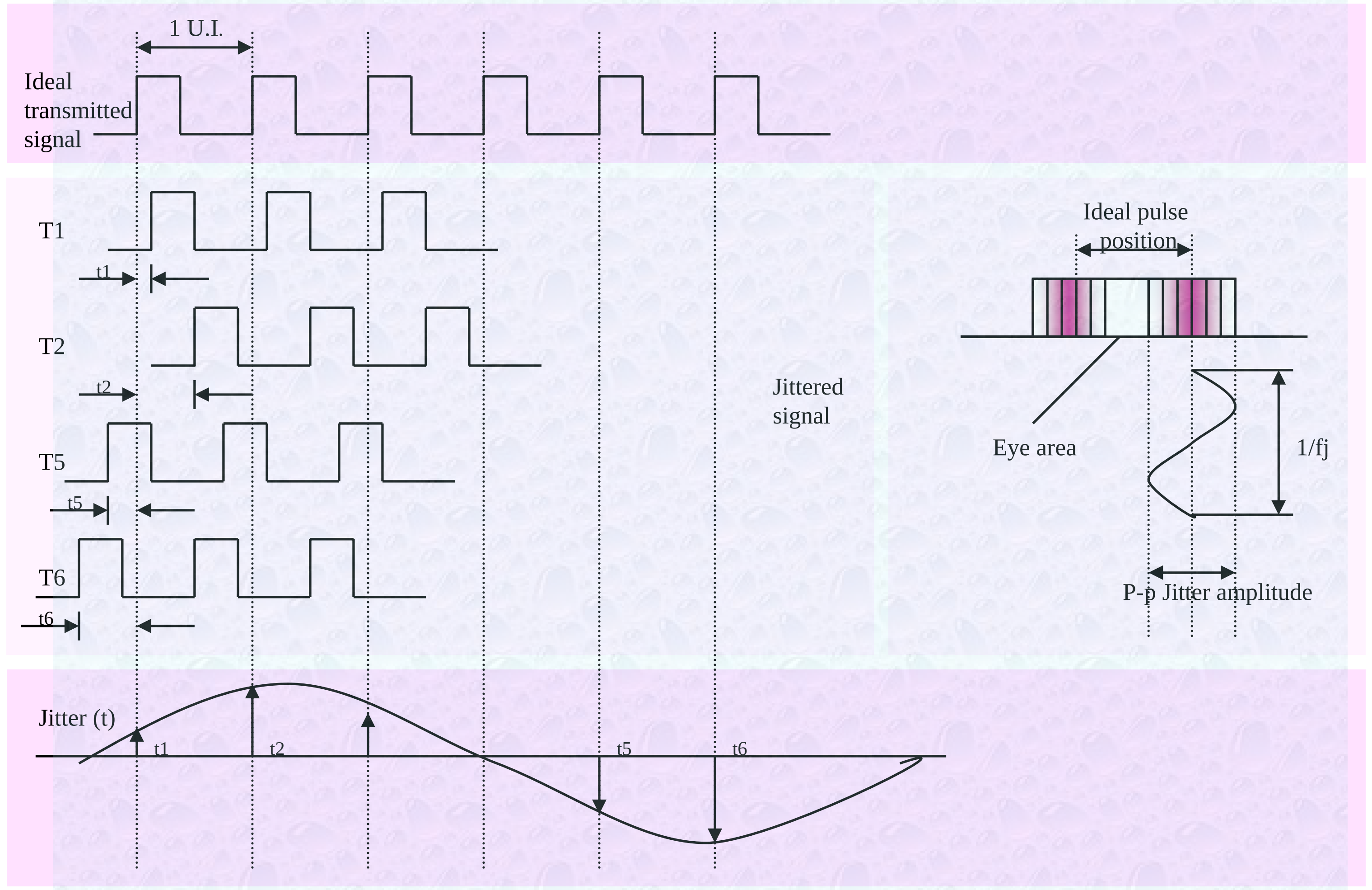
Jitter Measurements

Jitter generation : Jitter is added to ternary- or binary TX data or thrudata.

Modulation : - internal (1 Hz to 840 kHz)
- external via Ext Mod In port (0 Hz to 840 kHz)
- max. amplitude is 10.10 UI

Jitter analysis : - hit count, hit and hit free seconds or deciseconds
- jitter tolerance and jitter transfer function

Wireless Spread Spectrum
Timing jitter



XS40, XSP, and XS95 Board Manual

XESS Corporation

Limited Warranty

X Engineering Software Systems Corp. (XESS) warrants that the Product, in the course of its normal use, will be free from defects in material and workmanship for a period of one (1) year and will conform to XESS's specification therefor. This limited warranty shall commence on the date appearing on your purchase receipt.

XESS shall have no liability for any Product returned if XESS determines that the asserted defect a) is not present, b) cannot reasonably be rectified because of damage occurring before XESS receives the Product, or c) is attributable to misuse, improper installation, alteration, accident or mishandling while in your possession. Subject to the limitations specified above, your sole and exclusive warranty shall be, during the period of warranty specified above and at XESS's option, the repair or replacement of the product. The foregoing warranty of XESS shall extend to repaired or replaced Products for the balance of the applicable period of the original warranty or thirty (30) days from the date of shipment of a repaired or replaced Product, whichever is longer.

THE FOREGOING LIMITED WARRANTY IS XESS'S SOLE WARRANTY AND IS APPLICABLE ONLY TO PRODUCTS SOLD AS NEW. THE REMEDIES PROVIDED HEREIN ARE IN LIEU OF a) ANY AND ALL OTHER REMEDIES AND WARRANTIES, WHETHER EXPRESSED OR IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, AND b) ANY AND ALL OBLIGATIONS AND LIABILITIES OF XESS FOR DAMAGES INCLUDING, BUT NOT LIMITED TO ACCIDENTAL, CONSEQUENTIAL, OR SPECIAL DAMAGES, OR ANY FINANCIAL LOSS, LOST PROFITS OR EXPENSES, OR LOST DATA ARISING OUT OF OR IN CONNECTION WITH THE PURCHASE, USE OR PERFORMANCE OF THE PRODUCT, EVEN IF XESS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

In the United States, some statutes do not allow exclusion or limitations of incidental or consequential damages, so the limitations above may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Getting Help!

If you follow the instructions in this manual and you encounter problems, here are some places to get help:

- If you can't get the XS Board hardware to work, send an e-mail message describing your problem to `fpga-bugs@xess.com` or check our web site at `http://www.xess.com/FPGA`. Our web site also has
    - answers to frequently-asked-questions (`http://www.xess.com/FPGA/ho01000.html`),
    - example designs for the XS Boards (`http://www.xess.com/FPGA/ho03000.html`),
    - a place to sign-up for our email forum where you can post questions to other XS Board users (`http://www.xess.com/FPGA/list_reg.html`).
- If you can't get your XILINX F1 software tools installed properly, send an e-mail message describing your problem to `hotline@xilinx.com` or check their web site at `http://www.xilinx.com`.

1 Installing the XS40, XSP and XS95 Boards

1.1 WARNING!

The XS40, XSP and XS95 Boards require a power supply to operate! They do not draw power through the downloading cable from the PC parallel port.

1.2 Packing List

Here is what you should have received in your package:

- an XS40, XS95 or XSP Board (note that your XSP Board will be labeled as an XS40 but the socket will contain a Xilinx Spartan FPGA with an "XCS" prefix);
- a 6' cable with a male DB-25 connector on each end;
- a floppy diskette with software utilities for using the XS40, XSP or XS95 Board and documentation (you should be OK on this one).

1.3 Installing the XS Board Software Tools

XILINX currently provides the XACTstep F1 tools for programming their FPGAs and CPLDs. Any recent version of XILINX software should generate bitstream configuration files that are compatible with the XS40, XSP and XS95 Boards. Follow the directions XILINX provides for installing their software.

XESS Corp. provides the additional software utilities for interfacing the PC to the XS Board. Just activate the SETUP.EXE program on the 3.5" diskette to install these tools.

Once the additional software tools are installed, you will see the following subdirectories:

`XSTOOLS\BIN` contains the executable programs for downloading to the XS Board and for applying signals to the XS Board through the printer port. An assembler for the microcontroller is also included.

`XSTOOLS\DOCS` contains the documentation and schematics for the XS40, XSP and XS95 Boards.

1.4 Installing the XS40, XSP and XS95 Boards

1.4.1 Free-Standing Operation

You can use the XS Board all by itself to experiment with XC4000/XC9500 + 8031 designs. Just place the XS Board on a non-conducting surface. Then apply power to jack J9 of the XS Board from a 9V DC wall transformer with a 2.1 mm female, center-positive plug. The on-board voltage regulation circuitry will create the voltages required by the rest of the XS Board circuitry.

1.4.2 Protoboard Installation

The two rows of pins from the XS Board can be plugged into a protoboard with holes spaced at 0.1" intervals. (One of the A.C.E. protoboards from 3M is a good choice.) Once plugged in, all the pins of the XC4000/XC9500 and the 8031 microcontroller are accessible to other circuits on the protoboard. (The numbers printed next to the rows of pins on the XS Board correspond to the pin numbers of the XC4000 or XC9500.) Power can still be supplied to the XS Board though jack J9, or power can be applied directly through several pins of the XS Board. Just connect +5V and +3.3V to the following VCC pins of the XS40, XSP or XS95 Board, and connect ground to the GND pin. (You need +3.3V if your XS40 Board contains an XC4000XL type of FPGA.)

XS Board Type	GND Pin	+5V Pin	+3.3V Pin
XS95-072 V1.0	49	78	none
XS95-072 V1.1	49	78	none
XS95-072 V1.2	49	78	none
XS95-108 V1.0	49	78	none
XS95-108 V1.1	49	78	none
XS95-108 V1.2	49	78	none
XS40-005E V1.0	52	2	none
XS40-005E V1.1	52	2, 54	none
XS40-005E V1.2	52	2, 54	none

XS Board Type	GND Pin	+5V Pin	+3.3V Pin
XS40-005XL V1.0	52	none	54
XS40-005XL V1.1	52	2	54
XS40-005XL V1.2	52	2	54
XS40-010E V1.0	52	2	none
XS40-010E V1.1	52	2, 54	none
XS40-010E V1.2	52	2, 54	none
XS40-010XL V1.0	52	none	54
XS40-010XL V1.1	52	2	54
XS40-010XL V1.2	52	2	54
XSP-010	52	2,54	none

Table 1: Power and ground connections for all types and versions of the XS Boards.

1.5 XS Board-PC Connection

The 6' cable included with your XS Board connects it to the parallel port of your PC. One end of the cable attaches to the printer port and the other connects to the female DB-25 connector (J1) at the top of the XS Board as shown in Figure 1 (the XSP Board is identical with the XS40 Board V1.2).

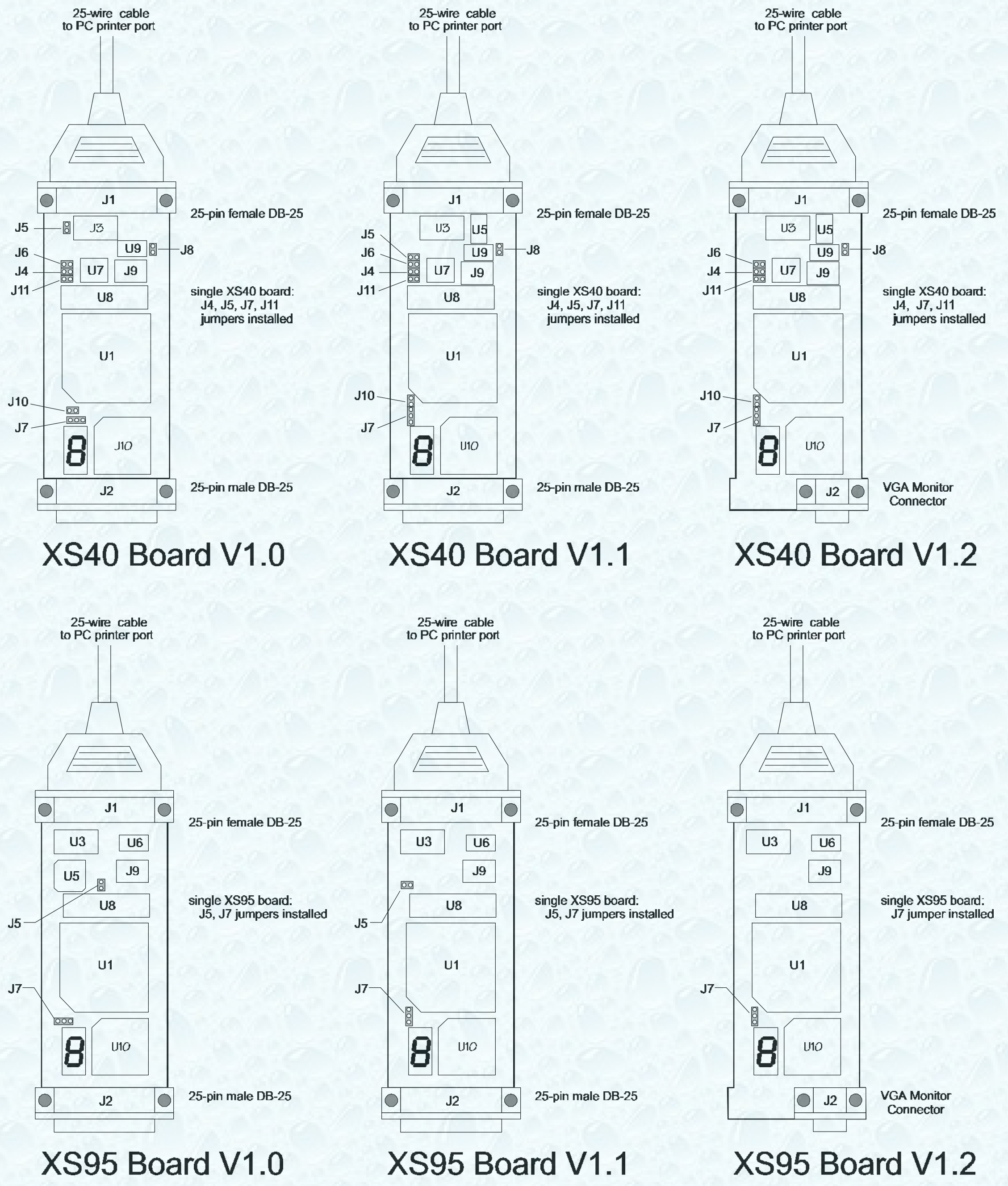

25-wire cable to PC printer port
J1
25-pin female DB-25
J5
J3
U9
J8
J6
J4
J11
U7
J9
U8
single XS40 board: J4, J5, J7, J11 jumpers installed
U1
J10
J7
U10
J2
25-pin male DB-25
XS40 Board V1.0
U3
U5
single XS40 board: J4, J5, J7, J11 jumpers installed
XS40 Board V1.1
single XS40 board: J4, J7, J11 jumpers installed
VGA Monitor Connector
XS40 Board V1.2
U6
single XS95 board: J5, J7 jumpers installed
XS95 Board V1.0
XS95 Board V1.1
single XS95 board: J7 jumper installed
XS95 Board V1.2

Figure 1: XS40/XSP/XS95-PC parallel port connection.

1.6 Configuring the XS40, XSP and XS95 Boards Jumpers

Jumper	Setting	Purpose
J4	**On (default)**	A shunt should be installed if you are downloading the XS40 or XSP Board through the parallel port.
	Off	The shunt should be removed if the XS40 or XSP Board is being configured from the on-board serial EEPROM (U7).
J5 (absent on V1.2 of XS40 or XSP)	**On (default)**	The shunt should be installed if you are using a single XS40 Board or if this is the last board in a cascaded chain of XS40 Boards.
	Off	The shunt should be removed on all but the last board in a chain of cascaded XS40 Boards.
J6	On	The shunt should be installed when the on-board serial EEPROM (U7) is being programmed.
	Off (default)	The shunt should be removed during normal board use.
J7	**1-2 (`ext`) (default)**	The shunt should be installed on pins 1 and 2 (`ext`) if the 8031 microcontroller program is stored in the external 32 KByte RAM (U8) of the XS40 Board.
	2-3 (`int`)	The shunt should be installed on pins 2 and 3 (`int`) if the program is stored internally in the 8031 chip.
J8	On	The shunt should be installed in XS40 or XSP Boards which use the 3.3V XC4000XL type of FPGAs.
	Off	The shunt should be removed on XS40 or XSP Boards which use the 5V XC4000E type of FPGAs.
J10	On	The shunt should be installed if the XS40 or XSP Board is being configured from the on-board serial EEPROM.
	Off (default)	The shunt should be removed if the XS40 or XSP Board is being downloaded from the PC parallel port.
J11	**On (default)**	The shunt should be installed if the XS40 or XSP Board is being downloaded from the PC parallel port.

Jumper	Setting	Purpose
	Off	The shunt should be removed if the XS40 or XSP Board is being configured from the on-board serial EEPROM.

Table 2: Jumper settings for the XS40 and XSP Boards.

Jumper	Setting	Purpose
J5 (absent on V1.2 of XS95)	**On (default)**	The shunt should be installed if you are using a single XS95 Board or if this is the last board in a cascaded chain of XS95 Boards.
	Off	The shunt should be removed on all but the last board in a chain of cascaded XS95 Boards.
J7	**1-2 (`ext`) (default)**	The shunt should be installed on pins 1 and 2 (`ext`) if the 8031 microcontroller program is stored in the external 32 KByte RAM (U8) of the XS95 Board.
	2-3 (`int`)	The shunt should be installed on pins 2 and 3 (`int`) if the program is stored internally in the 8031 chip.

Table 3: Jumper settings for the XS95 Board.

1.7 Testing the XS40, XSP and XS95 Boards

Once your XS Board is installed and the jumpers are in their default configuration, you can test the board using one of the following commands (you must be in the XSTOOLS\BIN directory to run the XSTEST command):

XS Board Type	Test Command
`XS95-072`	`XSTEST XS95-072`
`XS95-108`	`XSTEST XS95-108`
`XS40-005E`	`XSTEST XS40-005E`
`XS40-005XL`	`XSTEST XS40-005XL`
`XS40-010E`	`XSTEST XS40-010E`
`XS40-010XL`	`XSTEST XS40-010XL`
`XSP-010`	`XSTEST XSP-010`

Table 4: Commands for testing the various types of XS Boards.

The test procedure programs the FPGA or CPLD, loads the RAM with a test program for the microcontroller, and then the microcontroller executes this program. The total test period (including programming the board) is about 20 seconds for an XS40 or XSP Board, and about a minute for an XS95 Board. If the test completes successfully, then you will see a `O` displayed on the LED digit.

However, if the test program detects an error, then the LED digit displays an `E` or remains blank. In this case, check the following items:

- Make sure the board is receiving power from a 9V DC power supply through jack J9 or through the VCC and GND pins.
- Check that the board is sitting upon a non-conducting surface and that there are no connections to any of the pins (except for the VCC and GND pins if this is the way you are powering the board).
- Verify that the jumpers are in their default configuration.
- Make sure the downloading cable is securely attached to the XS Board and the PC parallel port.
- Verify that the parallel port is in SPP mode. (The mode is usually set in the BIOS as either SPP, EPP, or ECP. SPP is the safest and least ambitious mode.)

If all these checks are positive, then test the board using another PC. In our experience, 99.9% of all problems are due to the parallel port.

1.8 Programming the XS40, XSP and XS95 Boards

You can download an XC4000-based design into the XS40 or XSP Board as follows:

```
C:\> XSLOAD CIRCUIT.BIT
```

where `CIRCUIT.BIT` is an XC4000 bitstream file that contains the configuration for the XC4000 or XCS FPGA. Make sure the file contains a bitstream for the type of FPGA chip installed on your XS40 or XSP Board. This file is created using the XILINX F1 software tools.

You can download an XC9500-based design into the XS95 Board as follows:

```
C:\> XSLOAD CIRCUIT.SVF
```

where `CIRCUIT.SVF` is an XC9500 bitstream file that contains the configuration for the XC9500 CPLD. Make sure the file contains a bitstream for the type of XC9500 chip installed on your XS95 Board. This file is created using the XILINX F1 software tools.

Use one of the following commands if you need to configure the FPGA or CPLD and also download an Intel-formatted HEX file into the RAM of the XS Board:

```
C:\> XSLOAD FILE.HEX CIRCUIT.BIT

C:\> XSLOAD FILE.HEX CIRCUIT.SVF
```

XSLOAD assumes the XS Board is connected to parallel port #1 of your PC. You can use another port number like so:

```
XSLOAD -P 2 FILE.HEX CIRCUIT.BIT
```

1.9 Stand-Alone Configuration of the XS40, XSP and XS95 Boards

During the development and testing phases, you will usually connect the XS Board to the parallel port of a PC and download your circuit each time you make changes to it. But once your design is finished, you may want to store the design on the XS Board so that it is configured for operation as soon as power is applied.

This is easy with the XS95 Board. The XC9500 CPLD always stores its current configuration in an on-chip Flash memory. This configuration is restored whenever power is applied to the XS95 Board. So your design is always available even when the board is not connected to a PC.

But the XC4000 or XCS FPGA on the XS40 or XSP Board stores its configuration in an on-chip RAM which is erased whenever power is interrupted. However, an external serial EEPROM (such as the Atmel AT17C65/128/256) can be placed in socket U7 to store the FPGA configuration and reload it on power-up. You will have to perform several manual steps to 1) load the FPGA configuration into the EEPROM and 2) enable the configuration of the FPGA from the EEPROM.

Perform the following steps to load your design into the EEPROM:

1. Turn off power to the XS Board.
2. Place a shunt on jumper J6. This enables the programming circuitry in the Atmel EEPROM chip.
3. Apply power to the XS Board.
4. Use the following command to load the FPGA bitstream file into the EEPROM:

   ```
   C:\> XSLOAD –SERIAL_EEPROM CIRCUIT.BIT
   ```

5. Turn off power to the XS Board.
6. Remove the shunt on jumper J6. This disables the programming circuitry in the Atmel EEPROM chip so your design cannot be overwritten.

Once your design is loaded into the EEPROM, you must do the following to make the XS Board configure itself from the EEPROM instead of the PC parallel port interface:

1. Remove the downloading cable from connector J1 of the XS Board.
2. Place a shunt on jumper J10. This sets the FPGA into the active-serial mode so it will provide a clock signal to the EEPROM which sequences the loading of the configuration from the EEPROM into the FPGA.
3. Remove the shunt on jumper J4. This prevents the PC interface circuitry from interfering with the clock signal from the FPGA.
4. Remove the shunt on jumper J11. This prevents the PC interface circuitry from interfering with the data coming from the EEPROM.
5. Apply power to the XS Board. The FPGA will be configured from the serial EEPROM. You may reattach the downloading cable if you need to inject test signals into your design using the XSPORT program.

2 Designing with the XS40, XSP and XS95 Boards

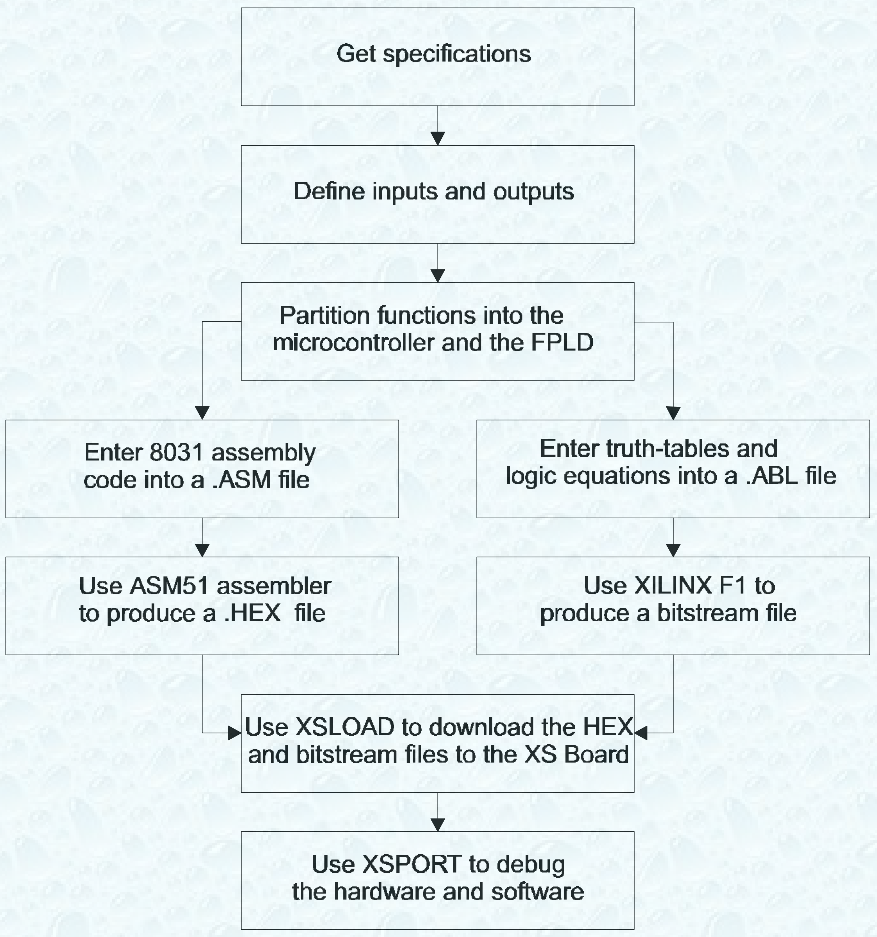
This section introduces the concepts required to create applications that use both the microcontroller and the FPLD (field programmable logic device). Building FPLD-based designs is covered in detail in the *Practical Xilinx Designer* by Prentice-Hall.

2.1 Microcontroller + FPLD Design Flow

The basic design flow for building microcontroller+FPLD applications is shown in Figure 2. Initially you have to get the specifications for the system you are trying to design. Then you have to determine what inputs are available to your system and what outputs it will generate.

At this point, you have to partition the functions of your system between the microcontroller and the FPLD. Some of the input signals will go to the microcontroller, some will go to the FPLD, and some will go to both. Likewise, some of the outputs will be computed by the microcontroller and some by the FPLD. There will also be some new intra-system inputs and outputs created by the need for the microcontroller and the FPLD to cooperate.

In general, the FPLD will be used mainly for low-level functions where signal transitions occur more frequently and the control logic is simpler. A specialized serial transmitter/receiver would be a good example. Conversely, the microcontroller will be used for higher-level functions where the responses occur less quickly and the control logic is more complex. Reacting to commands passed in by the receiver is a good example.



Get specifications
Define inputs and outputs
Partition functions into the microcontroller and the FPLD
Enter 8031 assembly code into a .ASM file
Enter truth-tables and logic equations into a .ABL file
Use ASM51 assembler to produce a .HEX file
Use XILINX F1 to produce a bitstream file
Use XSLOAD to download the HEX and bitstream files to the XS Board
Use XSPORT to debug the hardware and software

Figure 2: Microcontroller + FPLD design flow.

Once the design has been partitioned and you have assigned the various inputs, outputs, and functions to the microcontroller and the FPGA, then you can begin doing detailed design of the software and hardware. For the software, you can use your favorite editor to create a .ASM assembly-language file and assemble it with ASM51 to create a .HEX file for the 8031 microcontroller on the XS Board. For the FPLD hardware portion, you will enter truth-tables and logic equations into a .ABL file and compile it into a .BIT or .SVF bitstream file using the XILINX F1 programming software.

With the .HEX 8031-program file and the FPLD bitstream file in hand, you can download them to the XS Board using the XSLOAD program. XSLOAD stores the contents of the .HEX file into the 32 KByte RAM on the XS Board and then it reconfigures the FPLD by loading it with the bitstream file.

When the XS Board is loaded with the hardware and software, you need to test it to see if it really works. The answer usually starts as "No" so you need a method of injecting test signals and observing the results. XSPORT is a simple program that lets you send test signals to the XS Board through the PC parallel port. You can trace the reaction of your system to signals from the parallel port by programming the microcontroller and the

FPLD to output status information on the LED digit (much like placing "printf" statements in your C language programs). This is admittedly crude but will serve if you don't have access to programmable stimulus generators and logic analyzers.

2.2 Microcontroller+FPLD Interconnections

The 8031 microcontroller and the FPLD on the XS Board are already connected together. These existing connections save you the effort of having to wire them yourself, but they also impose limitations on how your program and the FPLD hardware will interact. A high-level view of how the microcontroller, RAM, and FPLD are connected is shown in
Figure **3**. More detailed schematics are presented at the end of this addendum.

The 12 MHz oscillator output goes directly to a synchronous clock input of the FPLD. The FPLD can control the clock it sends to the XTAL1 input of the microcontroller.

The 8031 multiplexes the lower eight bits of a memory address with eight bits of data and outputs this on its P0 port. Both the RAM data lines and the FPLD are connected to P0. The RAM uses this connection to send and receive data to and from the 8031. The FPLD is programmed to latch the address from P0 under control of the ALE signal and send the latched address bits to the lower eight address lines of the RAM.

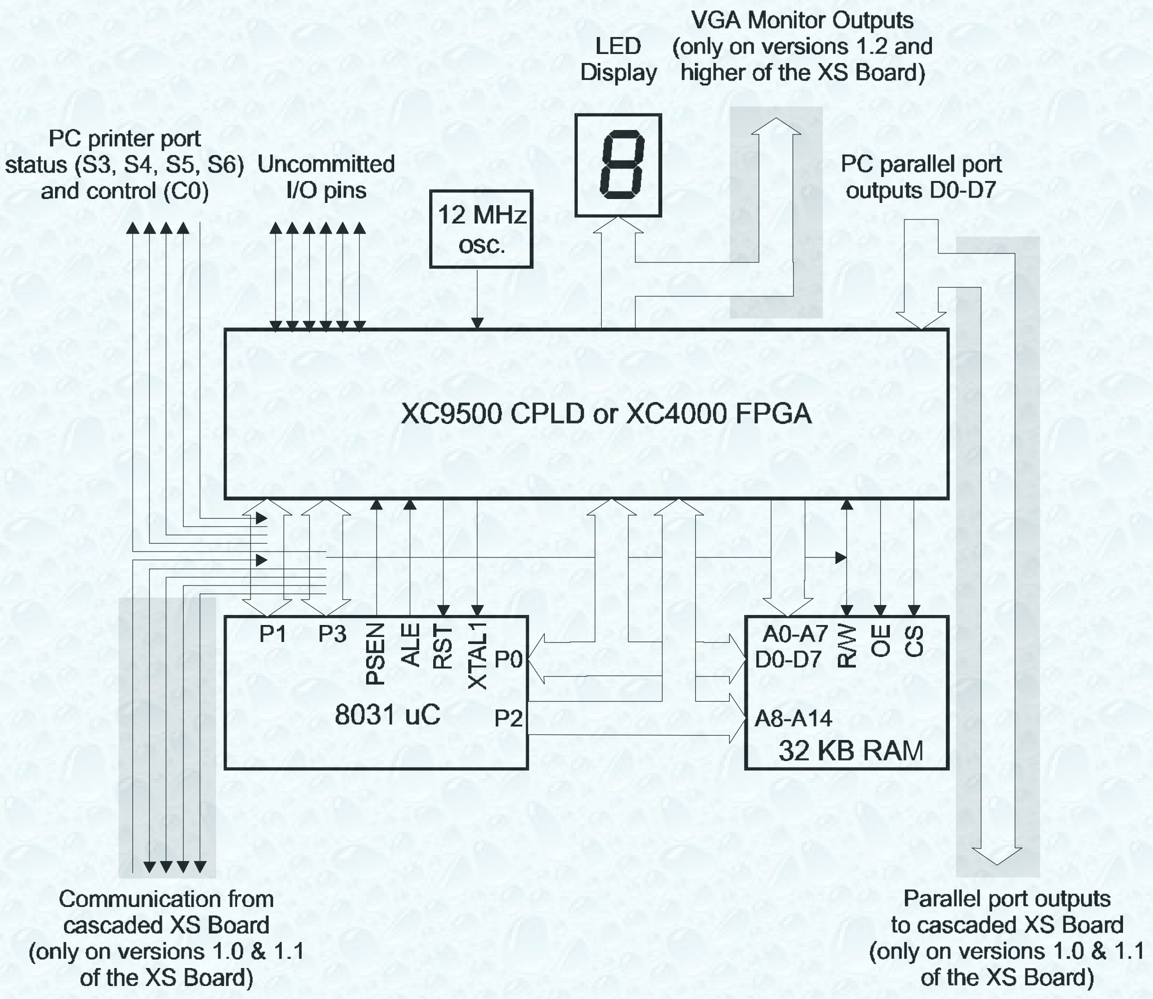

VGA Monitor Outputs (only on versions 1.2 and higher of the XS Board)
LED Display
PC printer port status (S3, S4, S5, S6) and control (C0)
Uncommitted I/O pins
12 MHz osc.
PC parallel port outputs D0-D7
XC9500 CPLD or XC4000 FPGA
P1
P3
PSEN
ALE
RST
XTAL1
P0
8031 uC
P2
A0-A7
D0-D7
R/W
OE
CS
A8-A14
32 KB RAM
Communication from cascaded XS Board (only on versions 1.0 & 1.1 of the XS Board)
Parallel port outputs to cascaded XS Board (only on versions 1.0 & 1.1 of the XS Board)

Figure 3: Connections between the 8031 microcontroller, RAM, and FPLD of the XS Board.

Meanwhile, the upper eight bits of the address are output on port P2 of the 8031. The RAM uses the lower seven of these address bits. The FPLD also receives the upper eight address bits and decodes these along with the PSEN and read/write control line (from pin P3.6 of port P3) from the 8031 to generate the CS and OE signals that enable the RAM and its output drivers, respectively. Either of the CS or OE signals can be pulled high to disable the RAM and prevent it from having any effect on the rest of the XS Board circuitry.

One of the outputs of the CPLD controls the reset line of the microcontroller. The 8031 can be prevented from having any effect on the rest of the circuitry by forcing the RST pin high through the FPLD. (When RST is active, most of the 8031 pins are weakly pulled high.)

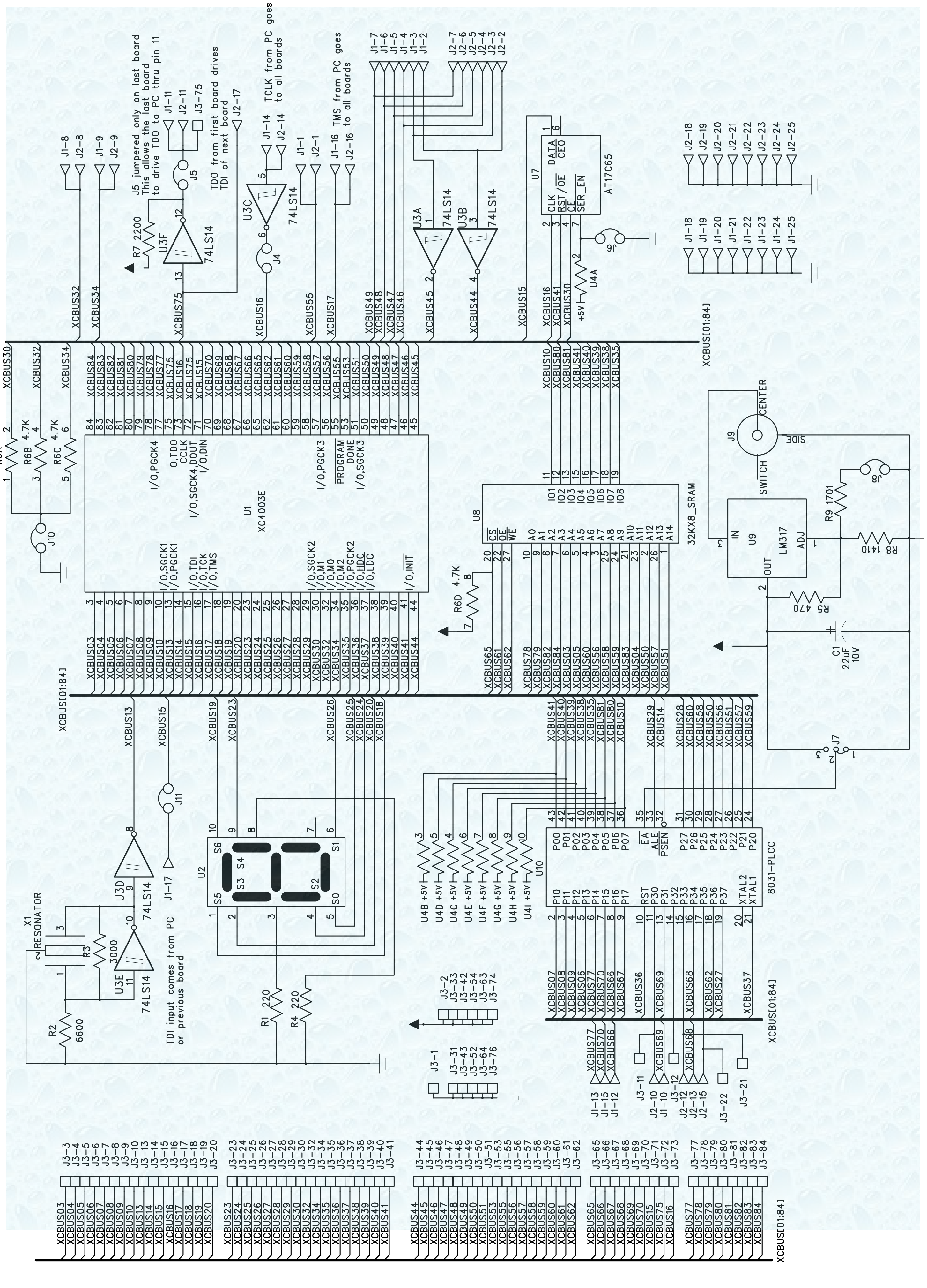
Many of the I/O pins of ports P1 and P3 of the 8031 connect to the FPLD and can be used for general-purpose I/O between the microcontroller and the FPLD. In addition to being general-purpose I/O, the P3 pins also have special functions such as serial transmitters, receivers, interrupt inputs, timer inputs, and external RAM read/write control signals. If you aren't using a particular special function, then you can use the associated pin for general-purpose I/O between the microcontroller and the FPLD. In many cases, however, you will program the FPLD to make use of the special-purpose 8031 pins. (For example, the FPLD could generate 8031 interrupts.) If you want to use the special-purpose pin with an external circuit, then the FPLD I/O pin connected to it must be tristated.

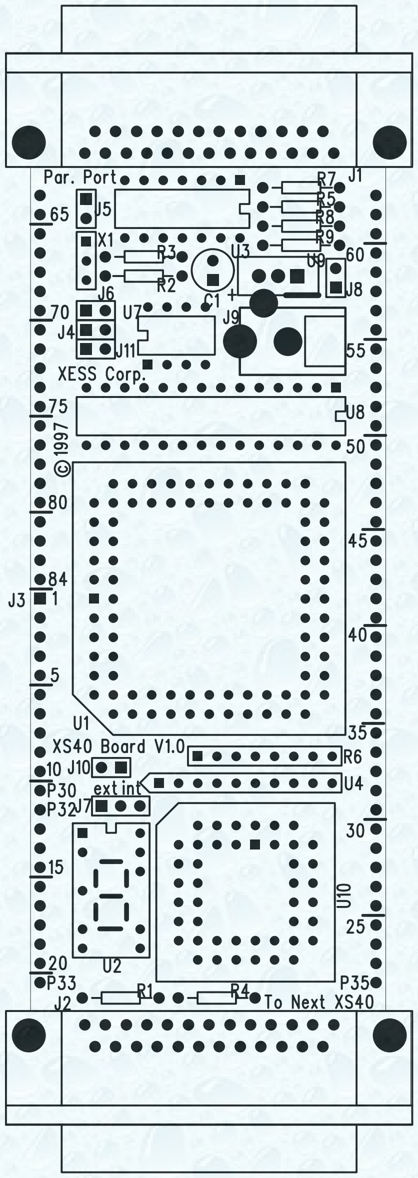
An LED digit connects directly to the FPLD. (These same FPLD pins also drive the VGA monitor connector on versions 1.2 and higher of the XS Board.) The FPLD can be programmed so the microcontroller can control the LEDs either through P1 or P3 or by memory-mapping a latch for the LED into the memory space of the 8031.

The PC can transmit signals to the XS Board through the eight data output bits of the printer port. The FPLD has direct access to these signals. The microcontroller can also access them by programming the FPLD to pass the data output bits onto the FPLD I/O pins connected to the 8031. The printer port data bits are also passed through the cascade header to the next XS Board in the chain (if there is one).

Communication from the XS Board back to the PC also occurs through the parallel port. Four of the parallel port status pins are connected to three pins of P1 and one pin of P3 . Either the microcontroller or the FPLD can drive the status pins. The PC can read the status pins to fetch data from the XS Board.

XS40 Board Version 1.0 Schematic

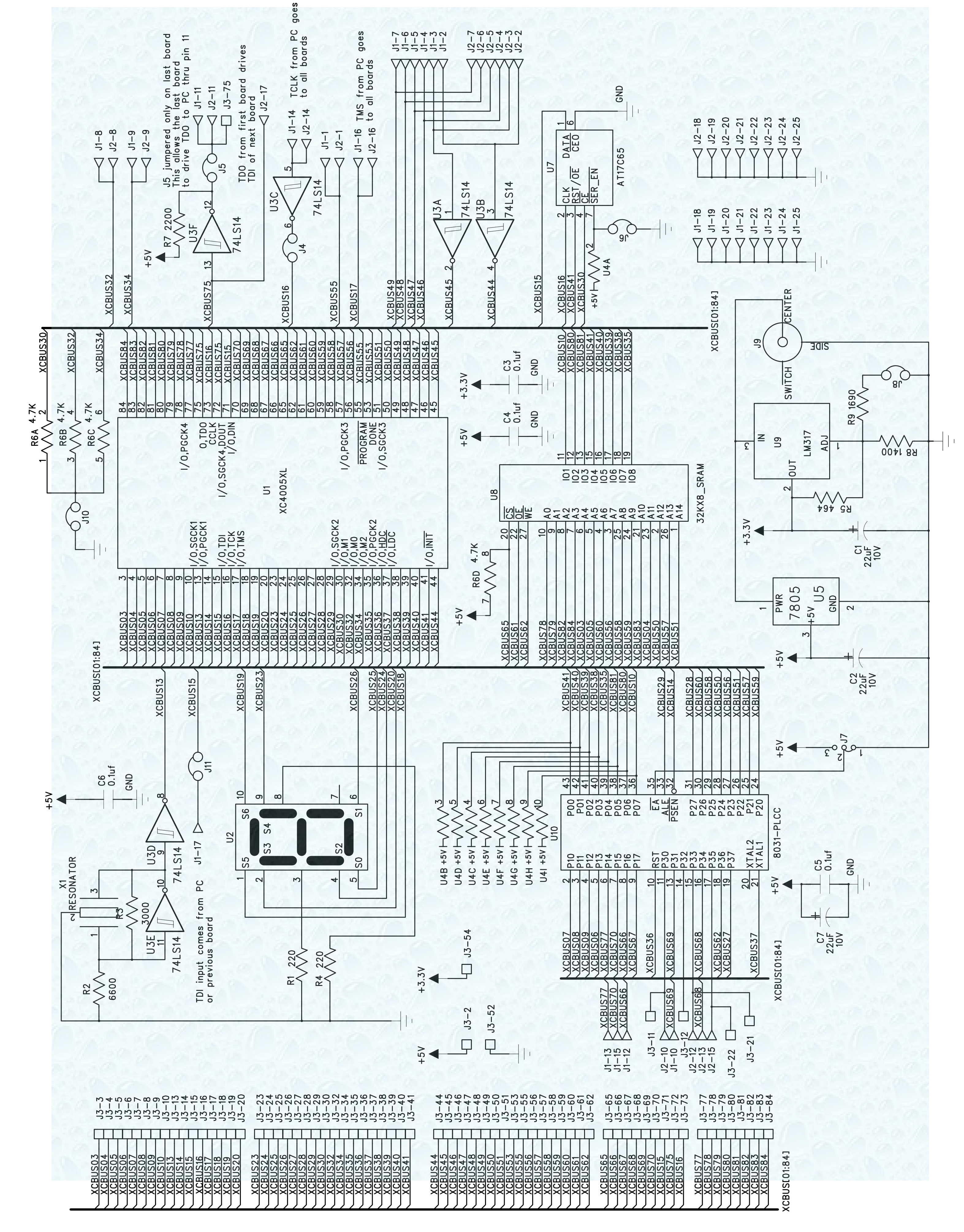




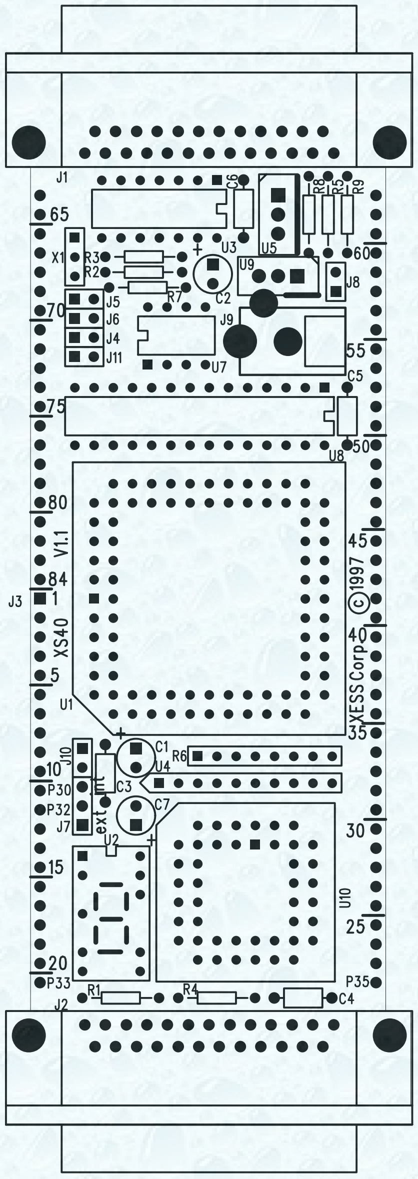
Layer 1 – TOP

Silkscreen (Top)

XS40 Board Version 1.1 Schematic



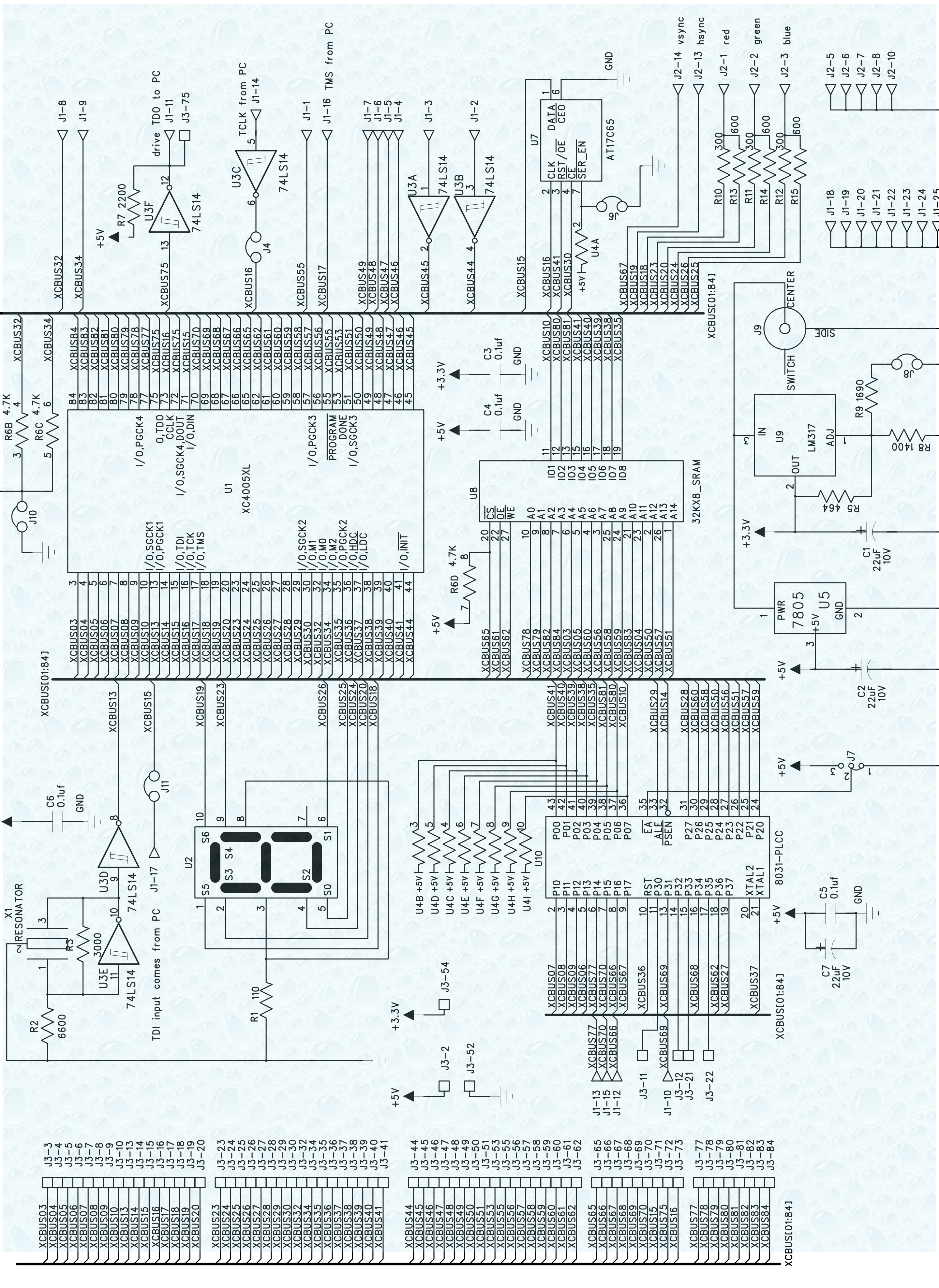
U1 XC4005XL
U8 32KX8_SRAM
U10 8031-PLCC
U7 AT17C65
U9 LM317
U5 7805
U2
X1 RESONATOR
U3A 74LS14
U3B 74LS14
U3C 74LS14
U3D 74LS14
U3E 74LS14
U3F 74LS14
J9 CENTER SWITCH SIDE
XCBUS[01:84]
J5 jumpered only on last board This allows the last board to drive TDO to PC thru pin 11
TDO from first board drives TDI of next board
TCLK from PC goes to all boards
TMS from PC goes to all boards
TDI input comes from PC or previous board
+5V
+3.3V
GND



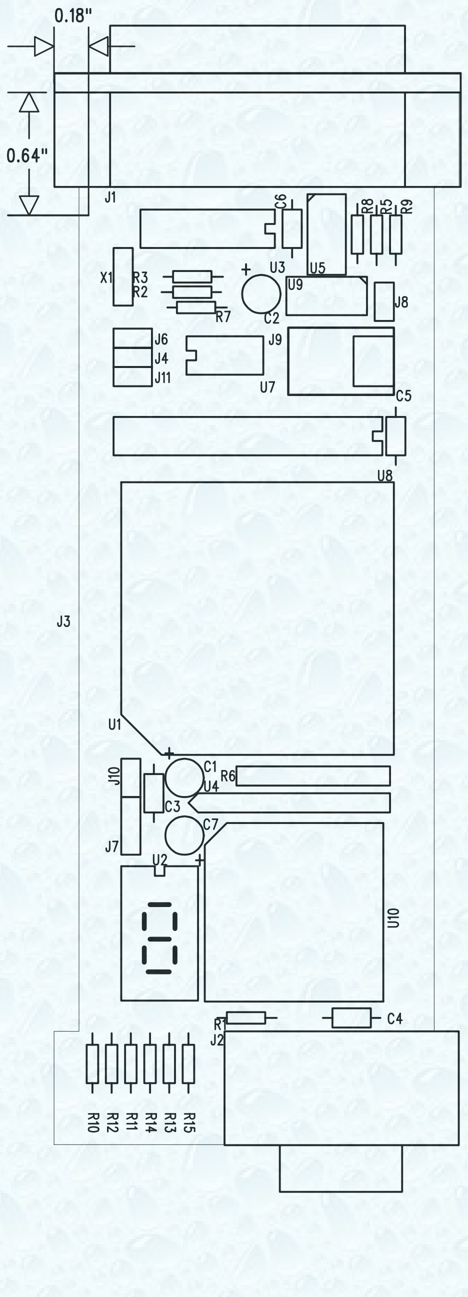
Layer 1 – TOP

Silkscreen (Top)

XS40 and XSP Board Version 1.2 Schematic



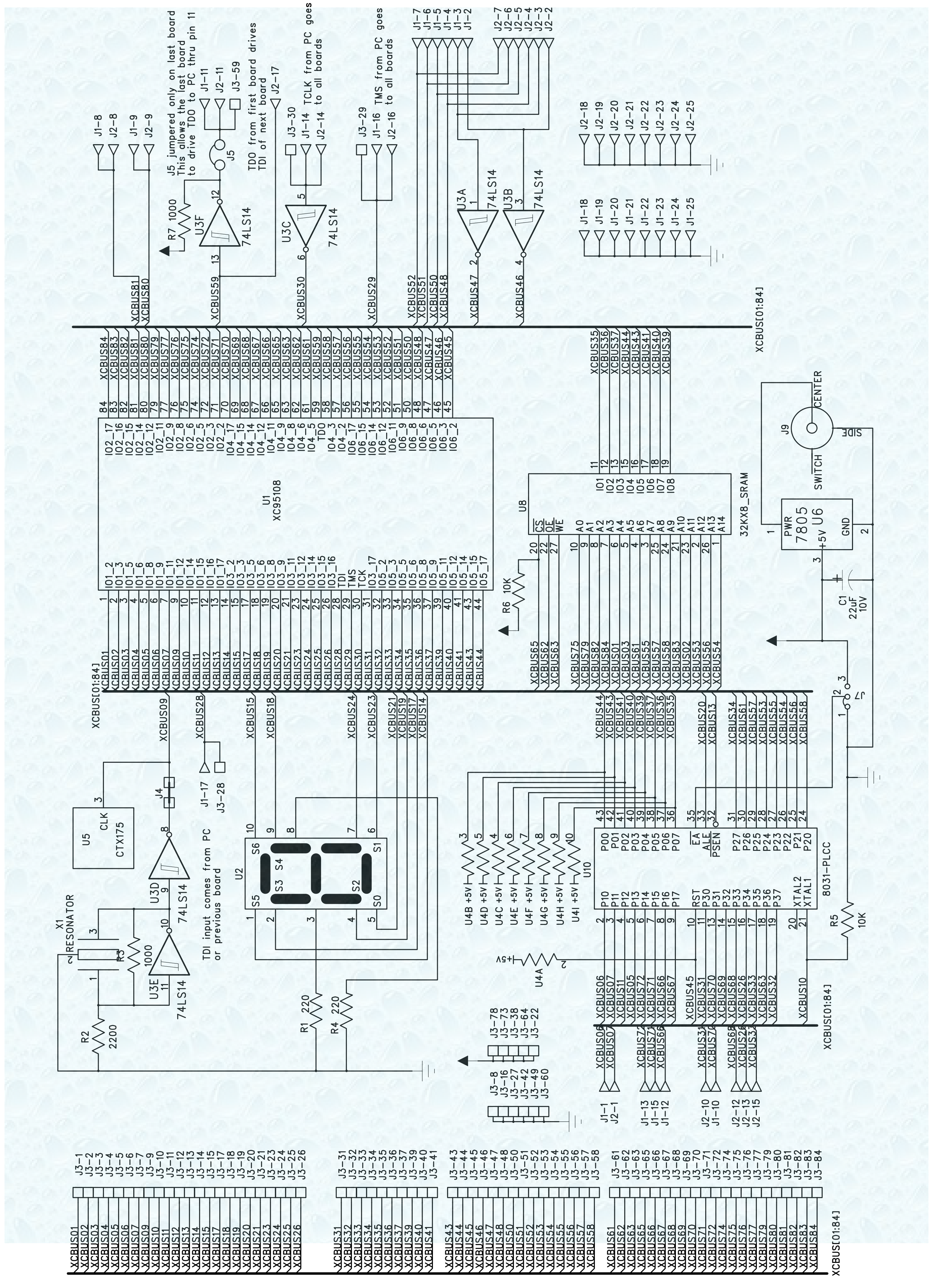
XS40 Board V1.2 Assembly Drawing - XESS Corp.

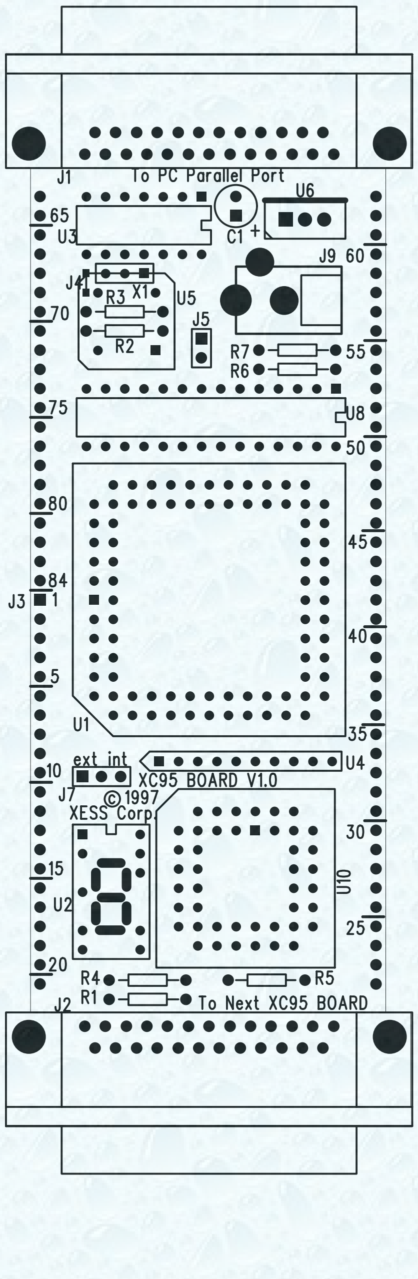


Layer 1 – TOP

Assembly Drawing (Top)

XS95 Board Version 1.0 Schematic

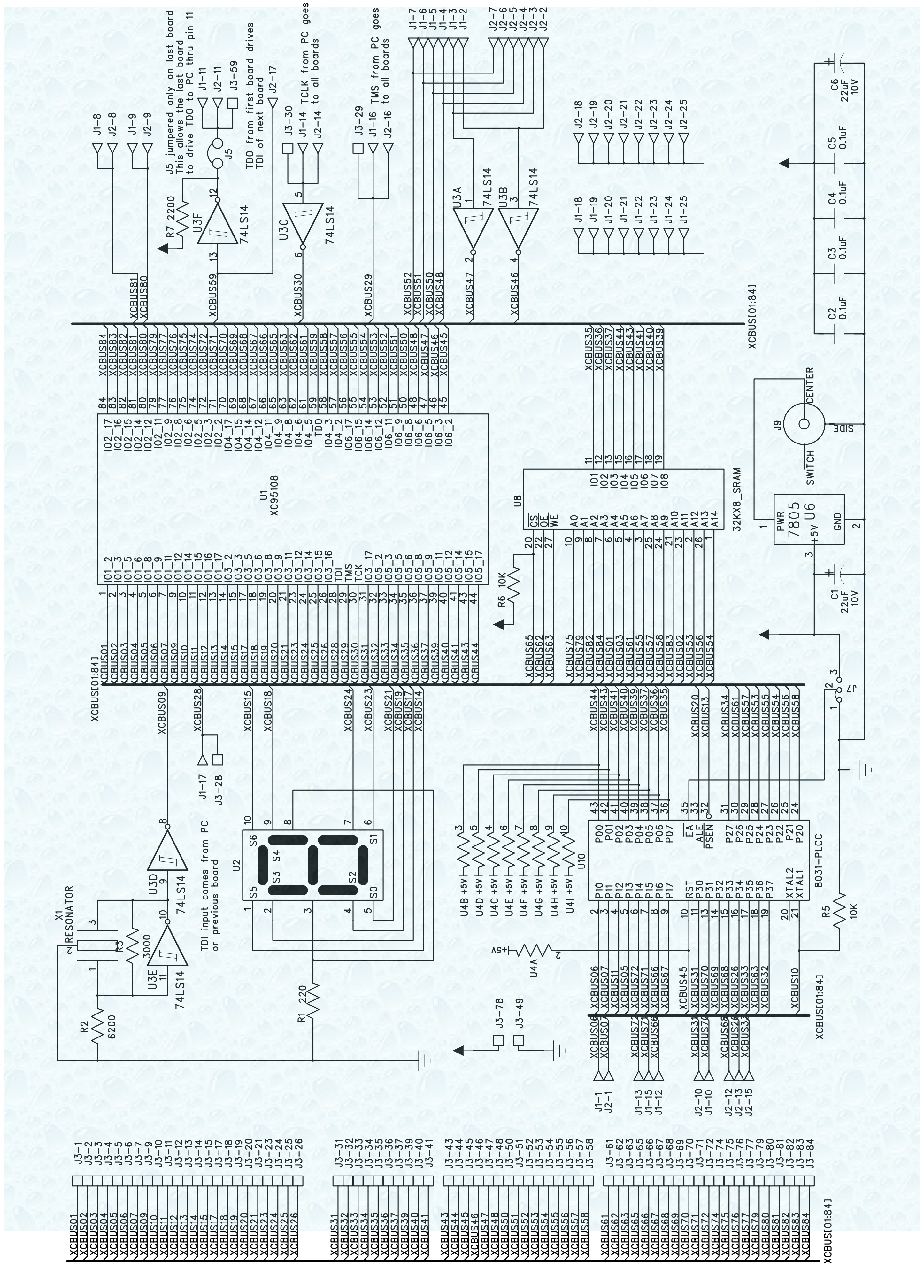




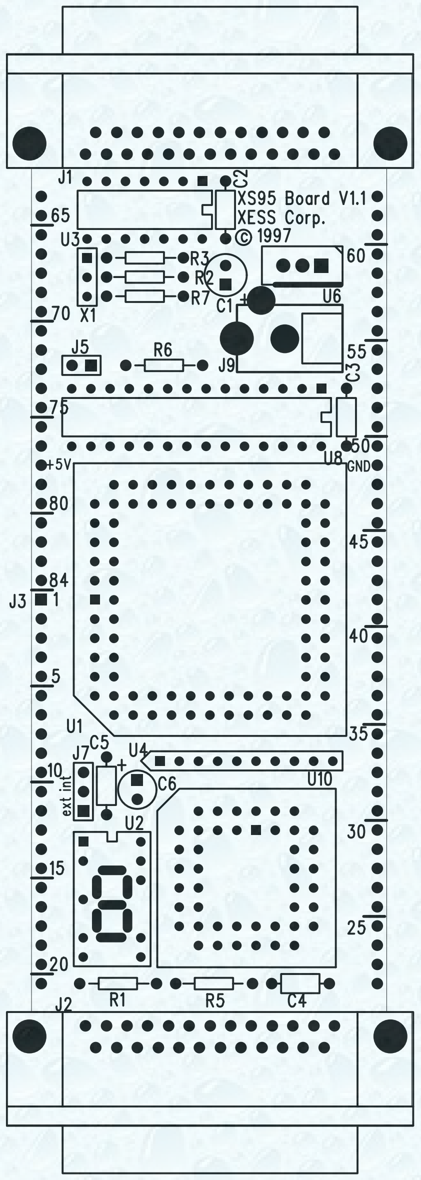
Layer 1 – TOP

Silkscreen (Top)

XS95 Board Version 1.1 Schematic



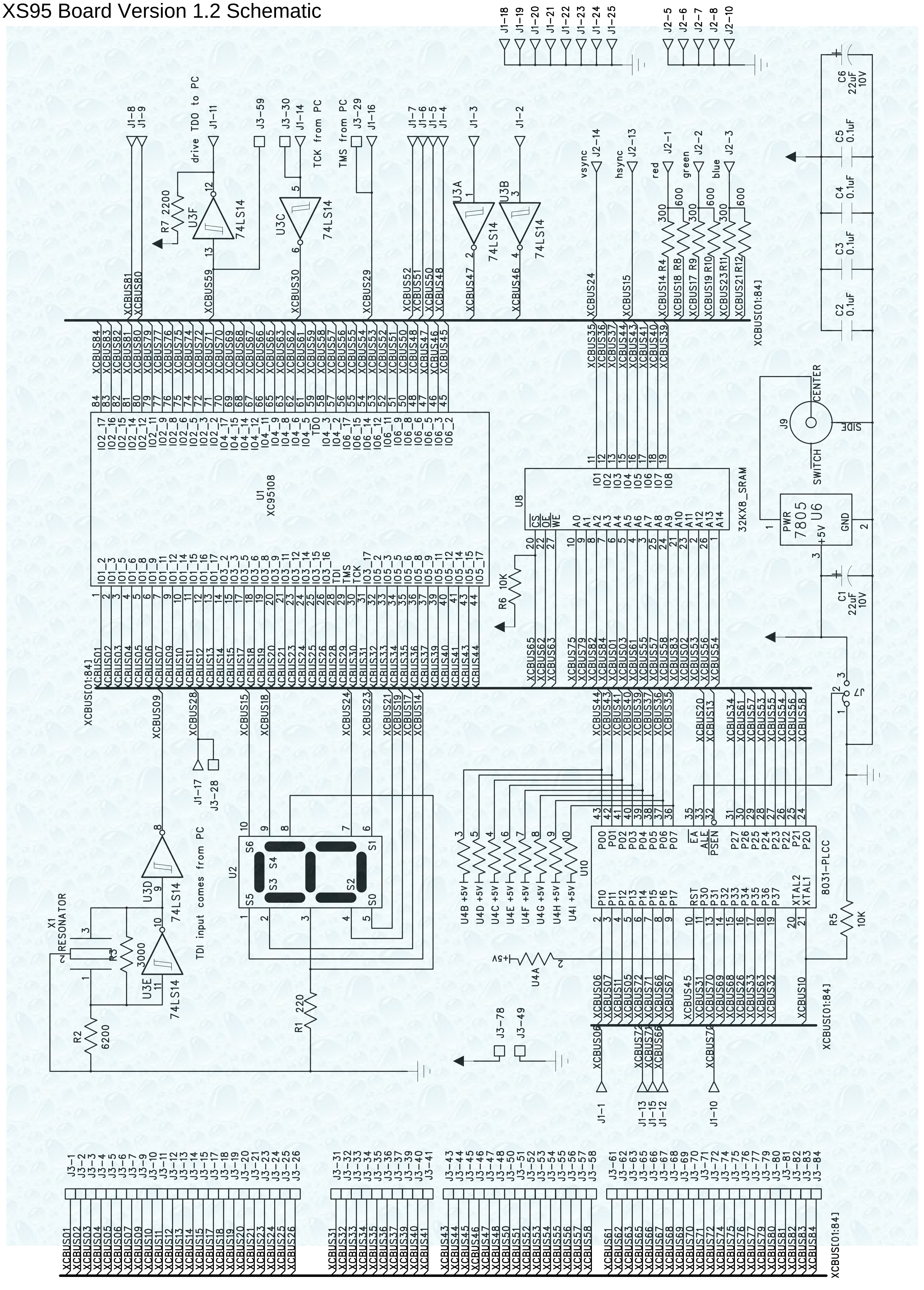
U1 XC95108
U8 32KX8_SRAM
U10 8031-PLCC
U6 7805
U2
J9
X1 RESONATOR
74LS14
J5 jumpered only on last board
This allows the last board to drive TDO to PC thru pin 11
TDO from first board drives TDI of next board
TCLK from PC goes to all boards
TMS from PC goes to all boards
TDI input comes from PC or previous board
XCBUS[01:84]
R1 220
R2 6200
R3 3000
R5 10K
R6 10K
R7 2200
C1 22uF 10V
C2 0.1uF
C3 0.1uF
C4 0.1uF
C5 0.1uF
C6 22uF 10V
PWR
GND
SWITCH
CENTER
SIDE



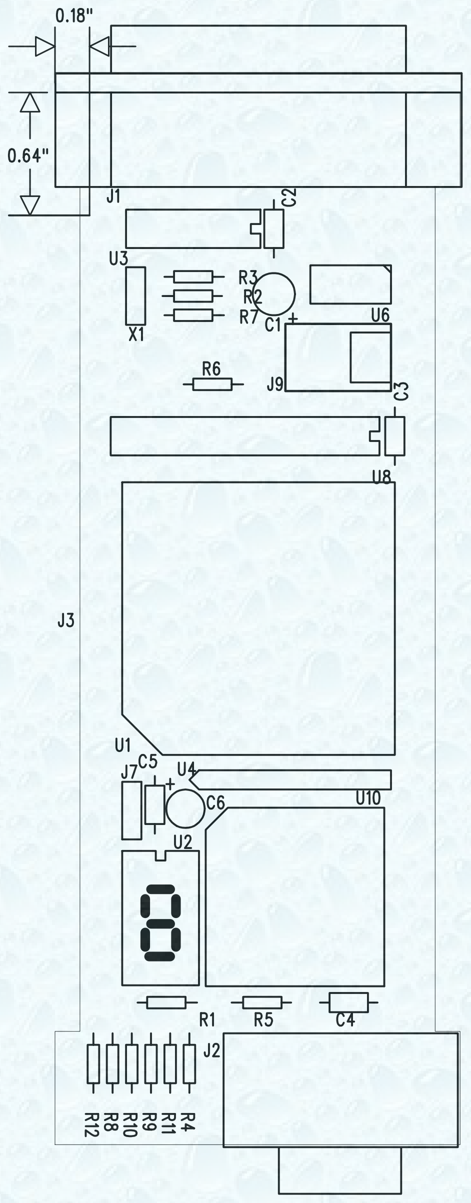
Layer 1 – TOP

Silkscreen (Top)

XS95 Board Version 1.2 Schematic

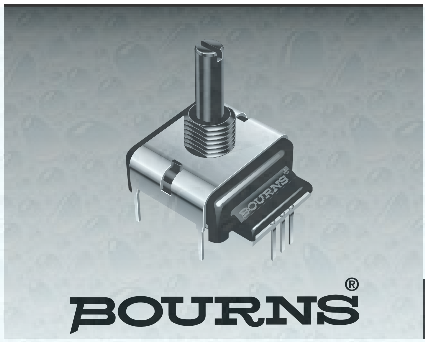


XS95 Board V1.2 Assembly Drawing - XESS Corp.



Layer 1 – TOP

Assembly Drawing (Top)



Features

- ■ Incremental encoder / quadrature output
- ■ Exceptionally long operating life
- ■ High operating temperature capabilities - up to 125°C
- ■ Sturdy construction
- ■ Bushing mount
- ■ Available with PC board mounting bracket (optional)

BOURNS®

ECW - Digital Contacting Encoder

Electrical Characteristics

Output	2-bit gray code, Channel A leads Channel B by 90° electrically turning clockwise (CW)
Closed Circuit Resistance	5 ohms maximum
Open Circuit Resistance	100K ohms minimum
Contact Rating	10 milliamp @ 10 VDC or 0.1 watt maximum
Insulation Resistance (500 VDC)	1,000 megohms minimum
Dielectric Withstanding Voltage	MIL-STD-202 Method 301
Sea Level	1,000 VAC minimum
Electrical Travel	Continuous
Contact Bounce (15 RPM)	5 milliseconds maximum
RPM (Operating)	120 maximum

Environmental Characteristics

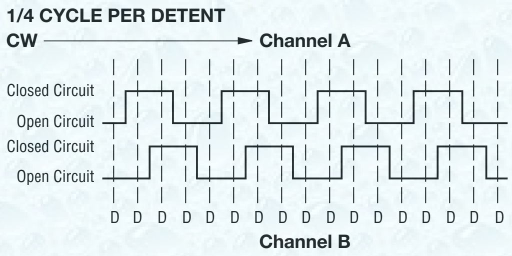
Storage Temperature Range	-40°C to +140°C
Operating Temperature Range	+1°C to +125°C
Humidity	MIL-STD-202, Method 103B, Condition B
Vibration	15G
Contact Bounce	0.1 millisecond maximum
Shock	50G
Contact Bounce	0.1 millisecond maximum
Rotational Life	200,000 shaft revolutions*

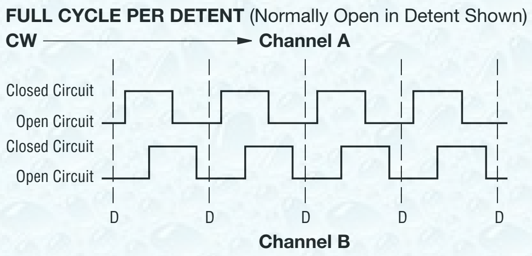
Mechanical Characteristics

Mechanical Angle	Continuous
Weight	Approximately 0.75 oz.
Torque (Detented)	0.75 to 2.25 oz-in.
Mounting Torque	7 in-lbs. maximum
Shaft Side Load (Static)	10 lbs. minimum

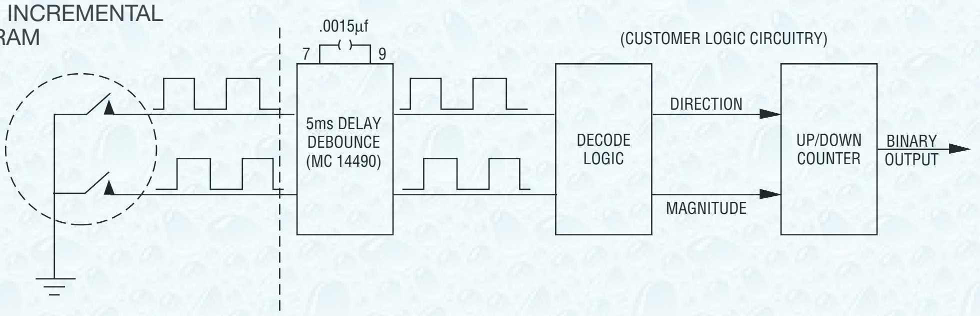
*Applies to EC Option.

QUADRATURE OUTPUT TABLE
This table is intended to show available outputs as currently defined.





RECOMMENDED INCREMENTAL CONTROL DIAGRAM



Specifications are subject to change without notice.

DIGITAL CONTACTING

The Digital Contacting Encoder is commonly referred to by such names as Digital Panel Control, Bit Switch, Gray Switch and Digital Switch. All such names are synonymous with a device whose output is a digital gray code signal, rather than a conventional potentiometric voltage ratio output.

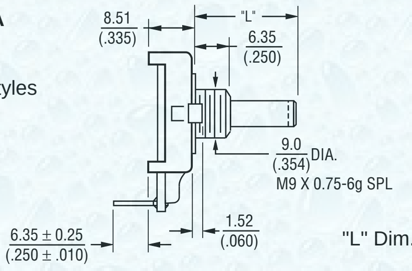
The advantage of the Digital Contacting Encoder is that it permits the direct entry of digitized analog data into a digital circuit without A/D conversion. The two (2) channel gray coded signal of this incremental encoder allows the user's decoder circuit to sense analog direction of rotation, as well as up-down counter capabilities . . . all without the time and cost required for A/D conversion. This approach can reduce memory overhead, wiring and wiring interconnects, and can provide greater MPU program speed.

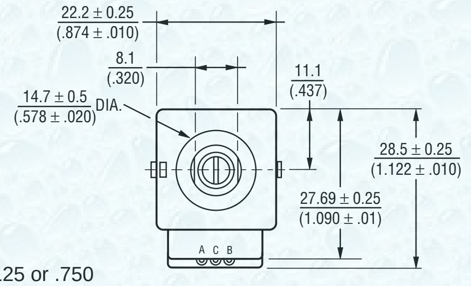
ECW - Digital Contacting Encoder

BOURNS®

BUSHING MOUNTED - HOUSING **A**
W style bushing shown.

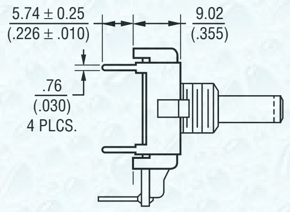
Shaft lengths "L" for B, C, R and Y styles
24 = .750" (19mm)
36 = 1.125" (28.5mm)

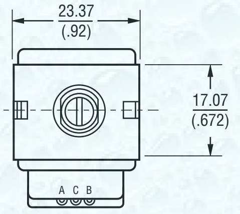




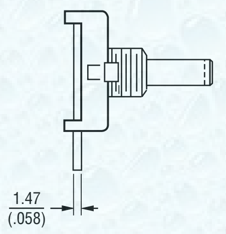
"L" Dim. = 1.125 or .750

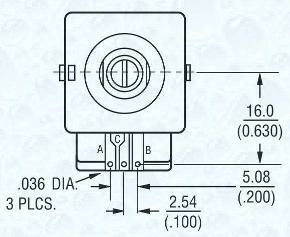
PCB BRACKET MOUNTED - HOUSING **B**
Dimensions not given are the same as Bushing Mounted.



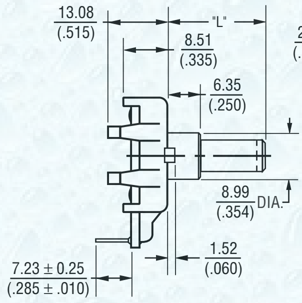


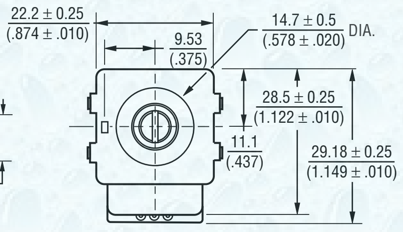
SOLDER HOLES - HOUSING **C**
Dimensions not given are the same as Bushing Mounted.

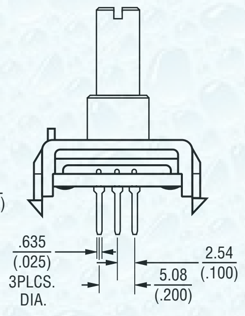




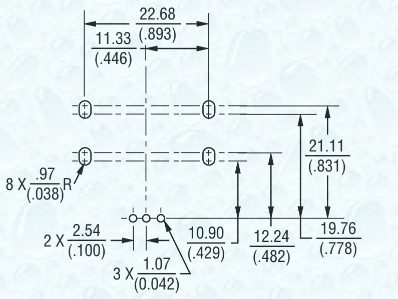
SNAP-IN MOUNT - Housing **G**



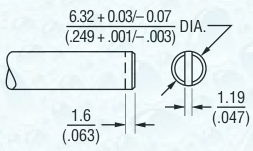




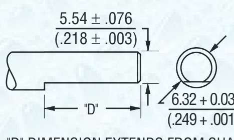
PCB MOUNTING DIMENSIONS



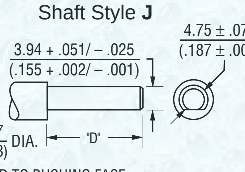
Shaft Style **B**



Shaft Style **C**

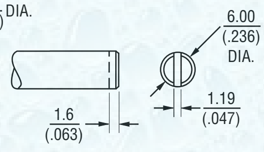


Shaft Style **J**

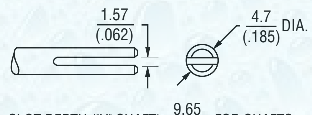


"D" DIMENSION EXTENDS FROM SHAFT END TO BUSHING FACE
"D" = (SHAFT LENGTH, FMS) – (BUSHING LENGTH)

Shaft Style **R**

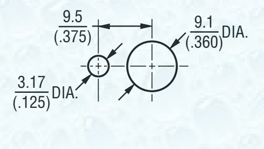


Shaft Style **Y**

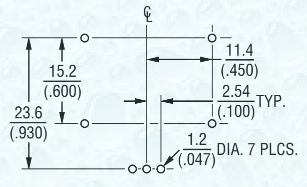


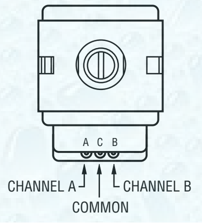
SLOT DEPTH ("Y" SHAFT) $\frac{9.65}{(3.80)}$ FOR SHAFTS < 1" LENGTH $\frac{14.7}{(5.20)}$ FOR SHAFTS ≥ 1" LENGTH

PANEL HOLE DIMENSIONS
Bushing Mounted



PCB MOUNTING DIMENSIONS
(Housing Styles B and E)





FOR TOLERANCES NOT SHOWN
.XX = ±.010
.XXX = ± .005
SHAFT DIMENSIONS ± 1/32"

DIMENSIONS ARE: $\frac{\text{METRIC}}{\text{(INCHES)}}$

ECW - Digital Contacting Encoder - How To Order

BOURNS®

PART NUMBERING SYSTEM

E	C	W	1	J	-	B	2	4	-	B	C	0	0	2	4

Code	Rotational Life
C	200,000 Revolutions

BUSHING CONFIGURATION

Code	Description
W	9mm x 1/4" Length. Threaded M9x0.75
L	9mm x 3/8" Length. Threaded M9x0.75 (Use B shaft only.)
T	9mm x 1/4". No Thread.

SWITCHING CONFIGURATION (In Detent Position)
Applies to performance codes B0012 and C0024 only, use code "0" for all other performance codes.

Code	Description
0	Not Applicable
1	Normally Open
2	Normally Closed

ANTI-ROTATION LUG POSITION

Code	Description
J	9:00 Position
D	None

SHAFT STYLE (See Outline Drawing for Details)

Code	Description
B	Plain with Inserted Slot (1/4" Dia.)
C	Single Flatted (1/4" Dia.)
R	Plain with Inserted Slot (6mm Dia.)
Y	Split Shaft Version (.185" Dia.)
J	Flatted Shaft (3/16" Dia.)

The sample part number demonstrates the identification code for Bourns contacting encoders.
The part number shown is a commonly used model, typically available from stock.

PERFORMANCE CODE

Code	Detents	Cycles/Rev.
E0006		6
E0009		9
E0012	0	12
E0024		24
B0012	12	12
C0006	24	6
C0024		24
D0009	36	9

HOUSING TERMINAL CONFIGURATION (X indicates "Equipped With"

	Code						
Features	**A**	**B**	**C**	**D**	**E**	**F**	**G***
Terminal Cover	X	X			X		X
Terminals	X	X			X		X
Solder Holes			X	X		X	
PCB Bracket		X		X	X	X	
Hardware Included	X		X		X	X	
Snap-In Mount							X

**Bushing code T only.*

SHAFT LENGTH (FMS)

Code	Description	Available Shaft Styles
16	1/2" Length	B
20	5/8" (15.9mm) Length	J
24	3/4" (19mm) Length	B, C, J, Y
28	7/8" (22.2mm) Length	B, C, J, Y
32	1" (25.4mm) Length	B, C, J, Y
36	1-1/8" (28.6mm) Length	B, C, J, Y
	Metric	
19	19mm Length	R
22	22mm Length	R
24	24mm Length	R

```
##### INPUTS
NET clk_12mhz          LOC=P13;
NET rst                LOC=P3;
NET spread_mode_sw     LOC=P4;
NET data_mode_sw       LOC=P5;
NET chb                LOC=P38;
NET cha                LOC=P35;

# PATTERN GENERATOR
NET sdo_prbs           LOC=P29;
#NET d6                LOC=P28;
#NET d5                LOC=P27;
#NET d4                LOC=P10;
#NET d3                LOC=P9;
#NET d2                LOC=P8;
#NET d1                LOC=P7;
#NET d0                LOC=P6;

##### OUTPUTS POD1
NET syn_cha_pin        LOC=P84;
NET syn_chb_pin        LOC=P83;
NET edgeup_pin         LOC=P82;
NET edgedo_pin         LOC=P81;
NET zin_pin            LOC=P80;
NET ld_pin             LOC=P79;
NET sh_pin             LOC=P78;
#NET d8                LOC=P67;
#NET d7                LOC=P65;
NET clk_1465hz_pin     LOC=P62;

##### OUTPUTS POD2
NET p_data_mode_pin        LOC=P61;
NET p_spread_mode_pin1     LOC=P60;
NET p_spread_mode_pin0     LOC=P59;
NET pn_start_pin           LOC=P58;
NET pn_ml1_pin             LOC=P57;
NET pn_ml2_pin             LOC=P56;
NET pn_gold_pin            LOC=P51;
NET sdo_spread             LOC=P50;
NET sdo_posenc_pin         LOC=P40;
#NET sdo_prbs              LOC=P39;

# 7-SEGMENT DISPLAY
NET seg_a                  LOC=P19;
NET seg_b                  LOC=P23;
NET seg_c                  LOC=P26;
NET seg_d                  LOC=P25;
NET seg_e                  LOC=P24;
NET seg_f                  LOC=P18;
NET seg_g                  LOC=P20;
```

```
##### INPUTS
NET clk_12mhz               LOC=P13;
NET rst                     LOC=P3;
NET spread_mode_sw          LOC=P4;
#NET data_mode_sw           LOC=P5;
NET sdi_spread              LOC=P38;
#NET chb                    LOC=P35;

# PATTERN GENERATOR
#NET d7             LOC=P29;
#NET d6             LOC=P28;
#NET d5             LOC=P27;
#NET d4             LOC=P10;
#NET d3             LOC=P9;
#NET d2             LOC=P8;
#NET d1             LOC=P7;
#NET d0             LOC=P6;

##### OUTPUTS POD1
NET semaphor_a_pin         LOC=P84;
NET semaphor_b_pin         LOC=P83;
NET semaphor_c_pin         LOC=P82;
NET semaphor_d_pin         LOC=P81;
NET semaphor_e_pin         LOC=P80;
NET extb_pin               LOC=P79;
NET chip_sample_pin        LOC=P78;
NET chip_sample1_pin       LOC=P67;
NET chip_sample2_pin       LOC=P65;
NET clk_23khz_pin          LOC=P62;

##### OUTPUTS POD2
NET seq_det_pin             LOC=P61;
NET spread_mode_pin1        LOC=P60;
NET spread_mode_pin0        LOC=P59;
NET full_sequence_pin       LOC=P58;
NET pn_ml1_pin              LOC=P57;
NET pn_ml2_pin              LOC=P56;
NET pn_gold_pin             LOC=P51;
NET pn_sequence_pin         LOC=P50;
NET bit_sample_pin          LOC=P40;
NET databit_pin             LOC=P39;

# 7-SEGMENT DISPLAY
NET seg_a_pin          LOC=P19;
NET seg_b_pin          LOC=P23;
NET seg_c_pin          LOC=P26;
NET seg_d_pin          LOC=P25;
NET seg_e_pin          LOC=P24;
NET seg_f_pin          LOC=P18;
NET seg_g_pin          LOC=P20;
```

```
PIN_NUMBER P13 .work.zender.behav.clk_12mhz
PIN_NUMBER P3 .work.zender.behav.rst
PIN_NUMBER P4 .work.zender.behav.spread_mode_sw
PIN_NUMBER P5 .work.zender.behav.data_mode_sw
PIN_NUMBER P38 .work.zender.behav.chb
PIN_NUMBER P35 .work.zender.behav.cha
PIN_NUMBER P29 .work.zender.behav.sdo_prbs
PIN_NUMBER P84 .work.zender.behav.syn_cha_pin
PIN_NUMBER P83 .work.zender.behav.syn_chb_pin
PIN_NUMBER P82 .work.zender.behav.edgeup_pin
PIN_NUMBER P81 .work.zender.behav.edgedo_pin
PIN_NUMBER P80 .work.zender.behav.zin_pin
PIN_NUMBER P79 .work.zender.behav.ld_pin
PIN_NUMBER P78 .work.zender.behav.sh_pin
PIN_NUMBER P62 .work.zender.behav.clk_1465hz_pin
PIN_NUMBER P61 .work.zender.behav.p_data_mode_pin
PIN_NUMBER P60 .work.zender.behav.p_spread_mode_pin1
PIN_NUMBER P59 .work.zender.behav.p_spread_mode_pin0
PIN_NUMBER P58 .work.zender.behav.pn_start_pin
PIN_NUMBER P57 .work.zender.behav.pn_ml1_pin
PIN_NUMBER P56 .work.zender.behav.pn_ml2_pin
PIN_NUMBER P51 .work.zender.behav.pn_gold_pin
PIN_NUMBER P50 .work.zender.behav.sdo_spread
PIN_NUMBER P40 .work.zender.behav.sdo_posenc_pin
PIN_NUMBER P19 .work.zender.behav.disp(0)
PIN_NUMBER P23 .work.zender.behav.disp(1)
PIN_NUMBER P26 .work.zender.behav.disp(2)
PIN_NUMBER P25 .work.zender.behav.disp(3)
PIN_NUMBER P24 .work.zender.behav.disp(4)
PIN_NUMBER P18 .work.zender.behav.disp(5)
PIN_NUMBER P20 .work.zender.behav.disp(6)
PIN_NUMBER P13 .work.RX.behav.clk
PIN_NUMBER P3 .work.RX.behav.rst
PIN_NUMBER P4 .work.RX.behav.switch
PIN_NUMBER P38 .work.RX.behav.sdi_spread
PIN_NUMBER P84 .work.RX.behav.sema(0)
PIN_NUMBER P83 .work.RX.behav.sema(1)
PIN_NUMBER P82 .work.RX.behav.sema(2)
PIN_NUMBER P81 .work.RX.behav.sema(3)
PIN_NUMBER P80 .work.RX.behav.sema(4)
PIN_NUMBER P79 .work.RX.behav.ex_tb
PIN_NUMBER P78 .work.RX.behav.chip_sample
PIN_NUMBER P67 .work.RX.behav.chip_sample1
PIN_NUMBER P65 .work.RX.behav.chip_sample2
PIN_NUMBER P62 .work.RX.behav.clk_2
PIN_NUMBER P61 .work.RX.behav.seq_det_pin
PIN_NUMBER P60 .work.RX.behav.spread_mode_pin1
PIN_NUMBER P59 .work.RX.behav.spread_mode_pin0
PIN_NUMBER P58 .work.RX.behav.full_seq_pin
PIN_NUMBER P57 .work.RX.behav.pn_ml1_pin
PIN_NUMBER P56 .work.RX.behav.pn_ml2_pin
PIN_NUMBER P51 .work.RX.behav.pn_gold_pin
PIN_NUMBER P50 .work.RX.behav.pn_sequence_pin
PIN_NUMBER P40 .work.RX.behav.bit_sample_pin
PIN_NUMBER P39 .work.RX.behav.databit_pin
PIN_NUMBER P19 .work.RX.behav.segdecoder(0)
PIN_NUMBER P23 .work.RX.behav.segdecoder(1)
PIN_NUMBER P26 .work.RX.behav.segdecoder(2)
PIN_NUMBER P25 .work.RX.behav.segdecoder(3)
PIN_NUMBER P24 .work.RX.behav.segdecoder(4)
PIN_NUMBER P18 .work.RX.behav.segdecoder(5)
PIN_NUMBER P20 .work.RX.behav.segdecoder(6)
```