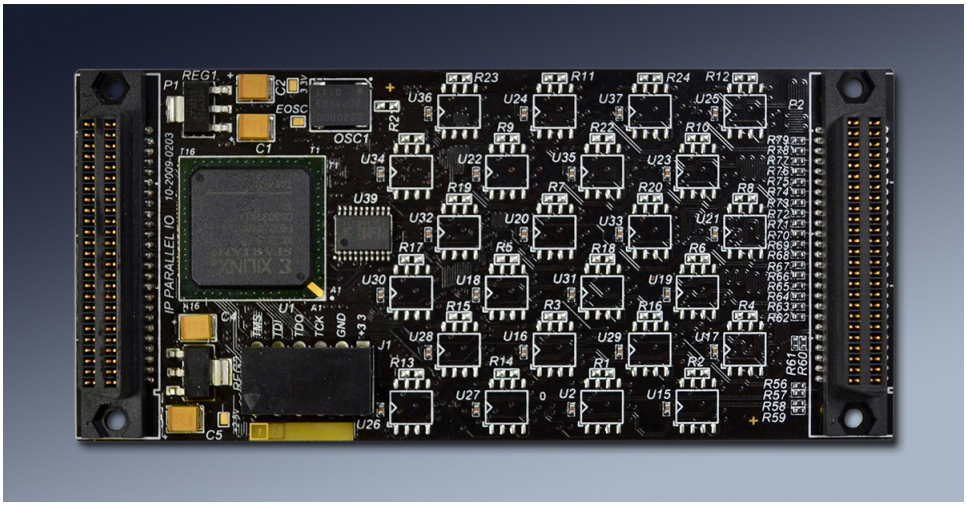
**DYNAMIC ENGINEERING**
150 DuBois St. Suite B/C Santa Cruz, CA 95060
831-457-8891
sales@dyneng.com
https://www.dyneng.com
Est. 1988

# User Manual

# IP-Parallel-TTL-PATT

## IP Module with Pattern Generation and GPIO

## IndustryPack® Module




Manual Revision: 1P1

Flash revision: 0101 Fab: 10-2001-0104

**IP-Parallel-TTL-PATT**
Pattern Generation plus GPIO
IndustryPack® Module


Revised 11/29/22

# Table of Contents

# List of Figures

## Product Description

IP-Parallel-TTL-PATT provides 48 IO with 32 allocated to Pattern Generation and 16 allocated to GPIO. Each GPIO is independent with programming options for direction, polarity, level or edge triggered with separate rising and falling enables. Two of the GPIO can be allocated to the Pattern Generation function to provide a reference clock and external trigger. The local 50 MHz oscillator is doubled to provide a 100 MHz reference. The reference is divided for the COS and pattern functions with separate programmable divisors. Filtered and direct IO access are also provided. Interrupts are programmable on a per line basis with 3 enables per line to cover the level, rising, and falling options.

Reference software for Win10 provide references for all of the modes of operation including setting up clocking, interrupts, using the parallel ports etc.

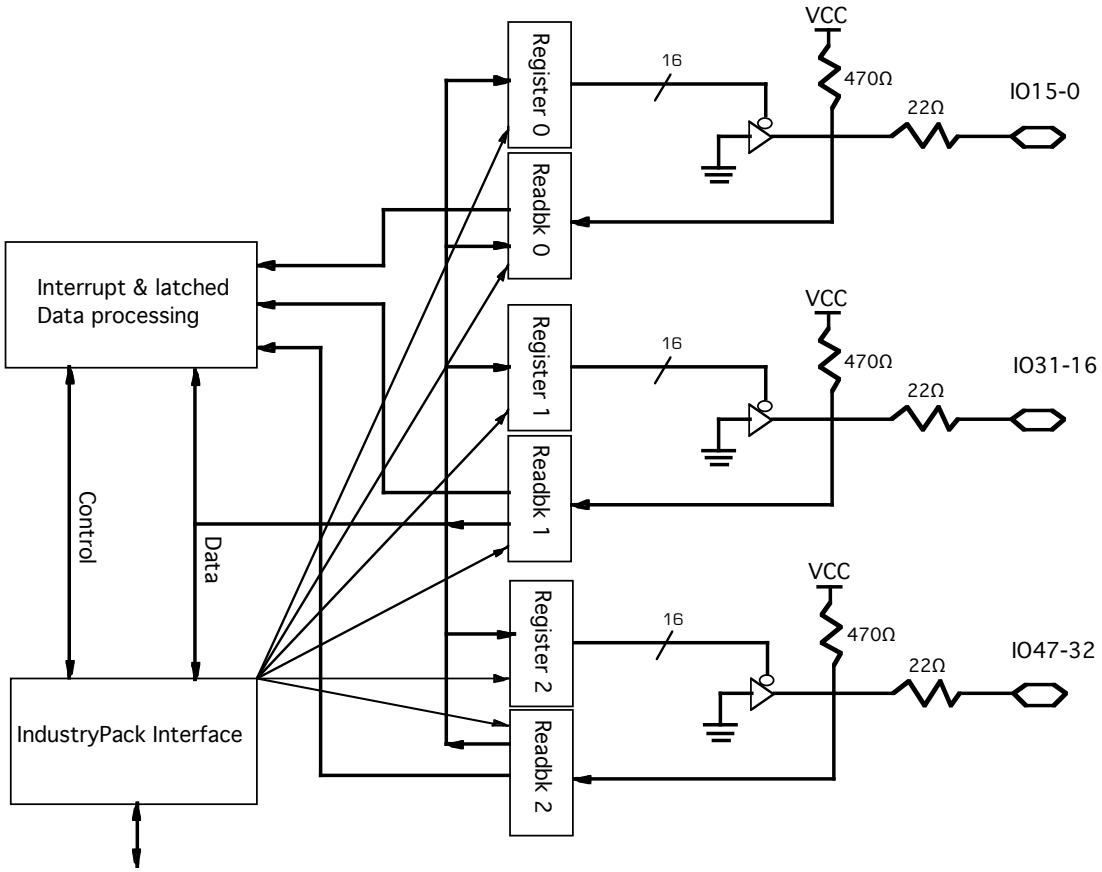



FIGURE 1 IP-PARALLEL-TTL-PATT BLOCK DIAGRAM

The Pattern interface uses IO 31-0 to provide 32 bits of output. User patterns stored in the Pattern FIFO are one option. Programmable options using counters etc. are provided for Ramp[rising, falling, rising-falling, falling-rising], square wave, Trapezoid, and Shift Up. User provided parameters allow the effective number of bits [out of the 32], as well as the number of times sent to be controlled.

The frequency of the pattern update is set by the 100 MHz reference divided by (N+1) where N is programmable. The state-machine has either 2 or 4 clocks per bit period depending on the mode selected. For example – N= 4, 100 MHz is divided by 5 for a 20 MHz reference. Selecting the Ramp Mode the bit period is 200 nS. The reference clock can be output on the top GPIO bit if desired. The clock provides a rising edge with ½ period set-up and hold. Programable option to invert the clock reference.

Another programmable option is the use of an external trigger. The COS section MSB-1 is used to trigger the pattern generator when enabled. Options for Rising or Falling, both or none. None indicates internal start based on Pattern Enable being set.

For the GPIO section, the COS N is used directly to control the sampling rate.

The 16 GPIO can be programmed to be transmitters or receivers with the direction selection. Data from the Tx register is driven onto the lines where the direction is set to Tx.

The received GPIO data can be read as Direct Data, Filtered Data, or COS status. Direct Data is the raw data after synchronization. Filtered Data has Polarity and EdgeLevel applied. Bits declared as edge are filtered out. Bits declared as inverted are reported after inversion. Active low signals on the IO can be read as active high in this manner.

The COS status is the captured change of state based on the settings in the enabled registers and the activity on the line. Separate Rising and Falling enables and status are supplied.

Interrupts are supported by GPIO. The interrupt occurs when the programmed event occurs. Status is provided in the Interrupt Status register – summary of Direct, COS Rising and COS Falling. The interrupts are individually maskable, and the interrupt vector is user programmable by a read/write register. The interrupt occurs on IntReq0. Status is available making it possible to operate in a polled mode. The COS interrupts are held until cleared by writing back to the respective status registers.

IP-Parallel-TTL-PATT is part of the IndustryPack® “IP” Module family of I/O components

by Dynamic Engineering. IP-Parallel-TTL-PATT provides an IP Module type II compliant mechanical package with a Spartan 6 FPGA, FLASH, 48 single ended IO each with separate direction controls. The FLASH is easily reprogrammable with the Xilinx Impact SW and USB adapter – a header is included for this purpose.

PATT is designed with a single level hierarchy – all logic at the Base level. The IP Bus interface and decoding, master interrupt control, clock generation, and Transmit and Receive logic.

IP-Parallel-TTL-PATT conforms to the VITA standard. This guarantees compatibility with multiple IP Carrier boards. Since the IP maintains plug and software compatibility while mounted on different form factors, system prototyping may be done on one IP Carrier board, with final system implementation done on a different one.

Currently about 60% of the FPGA is utilized. Room for more programmed modes. Send in your request with description of what the mode should do within the overall scheme described in this manual. We can either add it or spin off a new version depending on how much logic etc. are required to implement.

Currently implemented for the TTL version of IP-Parallel-IO. Let us know if you require a differential version. 24 IO available with IP-Parallel-IO outfit as 485 or LVDS. Could have 16 bit pattern with 8 bit GPIO or some other boundary.

## Product Details

IP-Parallel-TTL-PATT provides a Pattern Generator interface plus GPIO. The GPIO section features Change-Of-State capability. In addition, the Pattern Generator allocated IO can be used as a Parallel Port. All features are software programmable.

IP-Parallel-TTL-PATT features a Xilinx FPGA. The FPGA contains all of the registers and protocol controlling elements of the PATT design.

The Pattern interface is a specialized parallel port. IO 31-0 are allocated to the Pattern Generator. The Pattern Generator has 2 basic types- User or Pre-programmed. Type 0 is the User Data interface. Type1 -7 are pre-programmed.

For the user mode, data is written to the 4Kx32 FIFO and broadcast directly. The output rate is programmed with the divisor selected to use with the 100 MHz reference. With Win10 using the block move routines data can be continuously broadcast at 500 KHz. At 1 MHz we observed some small gaps where the FIFO emptied before the next set arrived. Likely some value higher than 500 KHz sustainable. Also with the reference clock output, a receiving device can filter out the gaps as the clock is stopped during the data outage. If broadcasting a pattern smaller than 4095 deep higher speeds can be selected as the whole pattern would fit within the FIFO.

For the rest of the programmed modes the data is internal and not dependent on the FIFO. The internal modes utilize registers to control operation. Start, Stop, Slope, Horizontal Count, Cycle Count, Mode, and the programmed frequency are used to select the shape of the waveform, number of times to repeat as well as the upper and lower limits.

For example, when the Trapezoidal shape is selected the pattern generated runs from Start to Stop, HCount at Stop, Stop – Start, HCount at Start. The rate of change from Start to Stop and Stop to Start is the slope. The end points are tested with >= and <= as appropriate to allow for Slopes that are not an even divisible of Stop – Start. Further the bounds are tested in HW to make sure not to over-run or under-run allowing 0 and xFFFF FFFF to be used for the end conditions. If the +/- Slope equation will produce an under-run or over-run situation the Start or Stop value is used instead. Rather than overshoot some rounding occurs at the numerical boundary. When away from the 0x00 and xFFFF FFFF boundaries the over / under shoot will occur allowing for simulation of ringing at the transition if desired.

Select Start = x10, Stop = x50, Slope = 5. x50-x40, Rising Ramp. The end conditions are inclusive. Output will be 0x10, 0x15, 0x1A … x4C, x51, 0x00. The end condition at

4C is not met and the sequence is not near the upper boundary. x51 is generated. With the slope set to 1 the end point in this case would be hit exactly.

The input signals are connected to the same IO bits as the output side [pattern generator]. The input side can operate in parallel with or separate from the transmit side. The input side of the pattern generator uses the Input clock to capture data. The clock is sampled with the 100 MHz reference and data sampled with the synchronized version. Signals up to 5 MHz can be captured. The data is stored into a second FIFO. The data can be read with single word reads or using the Multiple read call. See the SW manual for more information.

The Rx FIFO is supported with status, count, programmable almost full interrupt.

The Spartan VI requires three voltages, derived via Buck regulators from the 5V IP standard voltage [3.3V, 2.5V, and 1.2V]. The 3.3V rail is also used for the TTL devices allowing mix and match assembly. [RS-485 and LVDS with this PCB for other implementations]

The FLASH is used to store the module firmware – in this case the PATT and GPIO functions.

The Base level of the design provides the IndustryPack interface, general decoding, card level status, Vector register plus board level features, interrupt masking, master channel enable.

PATT and GPIO are implemented as a flat design with the Base and IO features at the same level. The decoded data bus is routed to each internal register to create control logic, status, data storage to support the serial IO and parallel ports.

Please refer the address/bit maps for details of the various registers and operation.

IP-Parallel-TTL-PATT conforms to the IndustryPack® standard. This guarantees compatibility with compliant carrier boards. Because IP-Parallel-TTL-PATT may be mounted on different form factors, while maintaining plug and software compatibility, system prototyping may be done on one Carrier board, with final system implementation on a different one. For example, the PCIe3IP – PCIe carrier for IP Modules can be used for development in a conventional PC. Later the hardware and software can be ported to the target. http://www.dyneng.com/PCIe3IP.html

With the Dynamic Engineering Windows driver collection for IP and carrier modules a Parent – Child architecture is employed. The IP portion of the driver is directly portable between the various Dynamic Engineering IP carriers [PCIe3IP, PCI3IP, PCIe5IP,

PCI5IP, PC104pIP, PC104p4IP, cPCI2IP, cPCI4IP etc]. The parent portion of the driver contains the carrier specific design information. This means that software developed for IP-Parallel-TTL-PATT on one platform can be directly ported to another. PCI to cPCI for example.

Designers can make use of the Dynamic Engineering carrier driver for non-Dynamic Engineering IP modules using the Generic IP capability built into the parent portion of the driver. IP modules that the carrier driver does not recognize are installed as "generic" and accessed with a address, data interface model. Software developed for the Generic mode can also be ported between modules.

IP-Parallel-TTL-PATT is tested with a combination of internal and external tests. The registers can be checked with R/W tests, the functions can be tested with the external loop-back fixture.

Interrupts are supported by IP-Parallel-TTL-PATT. A force interrupt for software development and test is provided, plus an interrupt from multiple received data options. Status is provided separate from the interrupt to allow use in a polled environment.

Change of State or "COS" is used to detect the transition of a signal from high to low or vice-versa. Each IO has a separate detector for the rising edge and falling edge. If both are enabled COS is implemented. Otherwise technically, it is a rising or falling detector. The transitions are based on the external signal. With an active low signal and falling edge enabled the leading edge of an active transition would be selected independent of the polarity setting. (polarity is only applied to filtered data).

When the edge is detected, the corresponding status bit is set and remains set until explicitly cleared by writing to the status register with the corresponding bit set. Reset also clears stored status.

If the interrupt for active bit is enabled, the ANDed enable and status are ORed together to create the rising or falling interrupt. If not enabled the status is available for polling. Please note the Master Interrupt Enable is also required for any of the programmed interrupts to create an interrupt to the host.

Level based interrupts are masked with the same interrupt enable bit and masked with the EdgeLevel selection. The Filtered data is the same data ahead of the interrupt enable being applied. Level data can also be polled in this manner. For level data the state of the IO line would need to change to remove the interrupt. In many cases the master or IO line level interrupt enable will need to be disabled until the line is in the ready state rather than the triggered state.

# Address Maps

## Address Map Base

| | |
|---|---|
| IpParTtlPATT_Base | 0x00 //0 Base control register |
| IpParTtlPATT_Vector | 0x02 //1 Interrupt vector register |
| IpParTtlPATT_IntStatus | 0x04 //2 Interrupt status read |
| IpParTtlPATT_Info | 0x06 //3 Spare port used by OS |
| IpParTtlPATT_Rev | 0x08 //4 Revision, Major&Minor, read only |
| IpParTtlPATT_PattMode | 0x0A //5 PATT Mode Control |
| IpParTtlPATT_PattFifoLo | 0x0C //6 Pattern FIFO lower 16 |
| IpParTtlPATT_PattFifoHi | 0x0E //7 Pattern FIFO upper 16 |
| IpParTtlPATT_PattDiv | 0x10 //8 Pattern Function Divisor N+1 |
| IpParTtlPATT_PattFifoCnt | 0x12 //9 Pattern FIFO Count |
| IpParTtlPATT_spare | 0x14 //10 unused |
| IpParTtlPATT_spare | 0x16 //11 unused |
| IpParTtlPATT_CosFallingEn | 0x18 //12 COS Falling Enable 15-0 |
| IpParTtlPATT_CosRisingEn | 0x1A //13 COS Rising Enable 15-0 |
| IpParTtlPATT_CosEdgeLevel | 0x1C //14 Edge or Level 15-0 |
| IpParTtlPATT_CosIntEn | 0x1E //15 Interrupt Enable 15-0 |
| IpParTtlPATT_CosPolarity | 0x20 //16 Polarity 15-0 |
| IpParTtlPATT_GpioDir | 0x22 //17 GPIO Direction 15-0 |
| IpParTtlPATT_GpioOut | 0x24 //18 GPIO Tx Data 15-0 |
| IpParTtlPATT_GpioDirect | 0x26 //19 Direct Read of IO 15-0 |
| IpParTtlPATT_HalfDiv | 0x28 //20 Set Divisor for COS Rate |
| IpParTtlPATT_CosRisingSt | 0x2A //21 COS Rising Status 15-0 |
| IpParTtlPATT_CosFallingSt | 0x2C //22 COS Falling Status 15-0 |
| IpParTtlPATT_CosFiltered | 0x2E //23 Filtered Read of IO 15-0 |
| IpParTtlPATT_Test0 | 0x30 //24 15-0 – R/W with TestIO set |
| IpParTtlPATT_Test1 | 0x32 //25 31-16 – IO bits. Read Always |
| IpParTtlPATT_StartLo | 0x34 //26 Pattern Start Register 15-0 |
| IpParTtlPATT_StartHi | 0x36 //27 Pattern Start Register 31-16 |
| IpParTtlPATT_StopLo | 0x38 //28 Pattern Stop Register 15-0 |
| IpParTtlPATT_StopHi | 0x3A //29 Pattern Stop Register 31-16 |
| IpParTtlPATT_CntLo | 0x3C //30 Pattern Count Register 15-0 |
| IpParTtlPATT_CntHi | 0x3E //31 Pattern Count Register 31-16 |
| IpParTtlPATT_RxFifoLo | 0x40 //32 Rx FIFO 15-0 |
| IpParTtlPATT_RxFifoHi | 0x42 //33 Rx FIFO 31-16 |
| IpParTtlPATT_RxFifoCnt | 0x44 //34 Rx FIFO Count |
| IpParTtlPATT_RxFifoAfl | 0x46 //35 Rx FIFO Almost Full Setting |
| IpParTtlPATT_SlopeLo | 0x48 //36 Pattern Slope Register 15-0 |
| IpParTtlPATT_HCountLo | 0x4C //38 Pattern Horizontal Count Register 15-0 |

FIGURE 2 IP-PARALLEL-TTL-PATT ADDRESS MAP

Numbers following the // are the HW decode numbers based on the words – word 0, word 1 etc. There are 64 available in the IP IO space.

## Programming

Programming IP-Parallel-TTL-PATT requires only the ability to read and write data from the host. The base address refers to the first user address for the slot in which the IP is installed.

The registers are organized as shown in the memory map and can be directly accessed with the offset to the IP Module position on the carrier. The base level has card level control and information. Channels when used, repeat with the same relative offsets to the start of each address range shown.

Depending on the software environment it may be necessary to set-up the system software with IP-Parallel-TTL-PATT “registration" data. Other OS may be more “plug and play”. The Dynamic Engineering Driver operates in a “plug and play” mode using parent ⇔ child architecture.

Interrupts are available to alert the local host when an event has happened.

The Dynamic Engineering drivers for Windows WFM (10, 11) and Linux, manage the interaction with the carrier level driver. Please refer to the driver manuals for more information. The manuals are downloadable from the webpage. Additional OS support can also be found on the website. Please contact Dynamic Engineering if you need additional SW options – VxWorks etc.

## Register Definitions

### IpParTtlPATT_Base

0x0000 // 0 base control register offset

| DATA BIT | DESCRIPTION |
|---|---|
| 15 | MasterIntEn |
| 14 | TestIOEn |
| 13 | LoopBackEn |
| 12 | ExtCapEn |
| 11 | ForceInt |
| 10 | Spare |
| 9 | RxFifoAflIntEn |
| 8 | PATT IntEn |
| 7-2 | Spare |
| 1 | PATT Reset |
| 0 | Spare |

FIGURE 3 IP-PARALLEL-TTL-PATT BASE CONTROL REGISTER BIT MAP

**PATT Reset** when '1' causes PATT state-machine to reset and clears associated FIFOs. '0' for standard operation.

**PATT Interrupt** when '1' used PATT state-machine status to cause interrupt request to host. Requires MasterIntEn to be enabled. Interrupt asserted at end of IO cycle. Most useful for read multiple commands.

**RxFifo AFL Interrupt** when '1' is enabled. The Latched status for Rx FIFO Almost Full condition is used to create the interrupt. Requires the Almost Full Level to be programmed. Requires MasterIntEn to be enabled. Interrupt asserted when FIFO has more than the programmed level.

**ForceInt**: when set causes an interrupt to be generated to the system. Requires MasterIntEn to be enabled. Useful for debugging and software test.

**LoopBackEn**: when set selects the loop-back path between the Pattern and Rx FIFOs. Cleared for normal operation. Data written to the Pattern FIFO is auto read and written to the Rx FIFO.

**ExtCapEn**: when set enables data capture into the Rx FIFO. The external clock input is

sampled and the rising edge used to capture data into the local FIFO. Leave disabled if not capturing data.

**TestIoEn**: when set selects the TEST0,1 registers to control the IO associated with the PATT function. Cleared for normal operation.

**MasterIntEn**: when set allows programmed interrupt sources to assert Int0. Required for all interrupt types. When '0' no interrupts will be requested, all status can be used for polling.

**IpParTtlPATT_VECTOR**
0x0002 // 2 IP vector port
Vector Port

| DATA BIT | DESCRIPTION |
|---|---|
| 15-8 | Undefined |
| 7-0 | vector |

FIGURE 4 IP-PARALLEL-TTL-PATT VECTOR BIT MAP

If the system uses a vectored interrupt approach the vector port should be initialized to the value assigned to this device. IP-Parallel-TTL-PATT can be used as vectored or auto-vectored. When auto-vectored this port is unused. The Status port can be used to determine the source of any pending interrupts from IP-Parallel-TTL-PATT.

Default is 0xFF for data.

**IpParTtlPATT_IntStatus**

0x0004 // 2 Interrupt Status register

| DATA BIT | DESCRIPTION |
|---|---|
| 15 | PattIdleState |
| 14 | Locked100 |
| 13 | IntReqM |
| 12 | IntReqUM |
| 11 | CosRisingIntReq |
| 10 | CosFallingIntReq |
| 9 | LvlIntReq |
| 8 | ForceInt |
| 7 | PatternFifoFull |
| 6 | PatternFifoMt |
| 5 | RxFifoFull |
| 4 | RxFifoMt |
| 3 | '0' |
| 2 | RxFifoAflLat |
| 1 | PatternEn |
| 0 | PattIntLat |

FIGURE 5 IP-PARALLEL-TTL-PATT INTERRUPT STATUS BIT MAP

**PattTIdleState:** When set '1' the PATT state-machine is in the idle state. When '0' the state-machine is processing commands.

**Locked100:** When '1' the local 100 MHz DCM has locked onto the 50 MHz [osc] reference and is generating the 100 MHz required for operation. The circuit has a watchdog reset control. If not locked within ~ 1mS the DCM is reset to retry locking on. If not set likely a problem with the oscillator.

**IntReqUM**: when set indicates an enabled interrupt request is pending. Signal level before the Master Interrupt Enable. "Unmasked"

**IntReqM**: When **IntReqUM** is set and the Master Interrupt Enable is set this bit will be set.

**CosRisingIntReq** when Set indicates at least one Rising Status bit is set along with the corresponding interrupt enable. When cleared no interrupts are pending from this source.

**CosFallingIntReq** when Set indicates at least one Falling Status bit is set along with

the corresponding interrupt enable. When cleared no interrupts are pending from this source.

**LvlIntReq** when Set indicates at least one bit declared as level, plus polarity, plus interrupt enable are true. When cleared no interrupts are pending from this source.

Note: COS interrupts are cleared by writing back to the Rising and or Falling status registers. Level interrupts are not cleared per se. Level interrupts are masked off until the IO line returns to the ready state, SW can then rearm the interrupt.

**ForceInt** is set when ForceInt is active in the base register. Slightly redundant having in both the base read and IntStatus read, and helpful for developing ISR routines to have the ForceInt behave as much like the IO interrupts as possible.

**PatternFifoFull:** is set when the Pattern FIFO is full. Should be '0' before loading a new command. 4095x32 are the dimensions of the Pattern FIFO.

**PatternFifoMt**: is set when the Pattern FIFO is empty. The Pattern function when enabled automatically reads from the Pattern FIFO when not empty.

**RxFifoFull**: is set when the Data Storage FIFO is at capacity [4095x32]. See Count register for direct read of current total.

**RxFifoMt**: when set the Data storage FIFO is empty.

**RxFifoAflLat**: When set the Rx FIFO level has exceeded the programmed value. The signal is captured and held. Cleared by write back with the same bit set. If the associated interrupt enable is set, the latched status is used to generate an interrupt request.

**PatternEn**: When set the Pattern enable bit in the mode control register is set. Duplicated here to incorporate into the status. When set the Pattern State Machine is enabled and could be executing a command. Check the Idle Status to see if the state machine is enabled but held in Idle or actually operating.

**PattIntLat:** Set when the Pattern interface completes a command. If the PattIntEn, and MasterInEn are also set an interrupt request can be generated.

**IpParTtlPATT_INFO**

0x0006 // 3 Location Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-11 | spare |
| 10-3 | Carrier Switch |
| 2-0 | Carrier Slot |

FIGURE 6 IP-PARALLEL-TTL-PATT INFO BIT MAP

The location register can be updated by the carrier driver during initialization. IP-Parallel-TTL-PATT Driver can access this information later to determine the carrier and location on the carrier that this node is installed into. Once the IP-Parallel-TTL-PATT Driver is started the user software can use this as a general purpose register. The IP Driver stores a local copy in RAM to allow the user software to determine which node it is talking to when multiple nodes are present in a PCI/PCIe based system with dynamic addressing. Please note that this function is supported on all Dynamic Engineering carriers and may not be supported on other products.

In a Windows system the user software will query for the installed devices and the devices will be returned in order. The issue is that the order can change and there is no way to directly tell with software which card you are controlling at the moment. The user software can retrieve the devices present and then match them up to the physical hardware based on the carrier switch setting and slot on the carrier.

In some systems, knowing which board is controlling which machine can be important. Please see the software manual and userap reference software for examples of working with multiple cards. The userap software prints out the device number and associated slot and switch settings. Your software can use the information without printing out for proper access control.

This is only important if you have multiple cards visible to the same CPU.

**IpParTtlPATT_REV**

0x0008 //4 Revision Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-8 | RevisionMajor |
| 7-0 | RevisionMinor |

FIGURE 7 IP-PARALLEL-TTL-PATT REVISION BIT MAP

RevisionMajor:
This field is reported via the IDPROM as well [revision]. It is rolled when major changes occur.

RevisionMinor:
This field is only read from this location. It is rolled when minor changes occur – usually during development to allow SW tracking of HW revisions without using the Major Revision field.

Revisions (Major P Minor)
1p1 original – New Design 11/22

**IpParTtlPATT_Mode**

x0A // 5

| DATA BIT | DESCRIPTION |
|---|---|
| 15 | ClockOut |
| 14 | ClockInvert |
| 13-12 | Trigger |
| 11 | Spare |
| 10 | Spare |
| 9 | AutoClearEn |
| 8 | Enable |
| 7-0 | TYPE |

FIGURE 8 IP-PARALLEL-TTL-PATT MODE DATA

**ClockOut** when set enables the reference clock corresponding to the output pattern on the top GPIO bit (47). When cleared the GPIO section has control of this bit.

**ClockInvert** when set inverts the ClockOut signal. Cleared for active high clock transition mid bit and set for falling edge mid bit.

**Trigger** is a 2 bit field used to select the trigger mode in use.
00 = no external trigger – start with Enable
01 = Rising edge trigger
10 = Falling edge trigger
11 = Both edge trigger

GPIO(46) is the input bit for the trigger. The COS clock reference affects detecting the transition – pick a rate fast enough to detect the transition. The Trigger is used along with the SW enable [need both]. Idle bit in the status register will be set until the trigger is detected. When changing trigger definitions, it is recommended to use the Base Register Reset to make sure the trigger has not already been captured. Do the set-up with the enables, etc. then reset. The local reset only affects the FIFOs and local state-machine operation. For Rising the Rising Enable for bit 14 needs to be set. For Falling the Falling Enable for bit 14 needs to be set. For both edges the Rising and Falling enables need to be set.

**AutoClearEn** when set causes the **Enable** control bit to be reset at the end of a sequence. For example if the programmed count is 10, after the 10$^{th}$ cycle the bit is cleared preventing the cycle from repeating when the pattern state machine returns to the Idle state.

**Enable** when set causes the pattern function to start if not further constrained by a trigger requirement. Can be cleared at the end of the programmed operation if **AutoClearEn** is set.

**Type** sets the operation to execute when enabled.
Type 0 = User Pattern
Type 1 = Rising Ramp
Type 2 = Falling Ramp
Type 3 = Pyramid
Type 4 = Inverted Pyramid
Type 5 = SquareWave
Type 6 = Trapezoidal
Type 7 = Shift Up
See the Type description section for more detail.

**IpParTtlPATT_PatternFifoLo**

x0C // 6

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | 15-0 |

FIGURE 9 IP-PARALLEL-TTL-PATT PATTERN FIFO LO

**IpParTtlPATT_PatternFifoHi**

x0E // 7

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | 31-16 |

FIGURE 10 IP-PARALLEL-TTL-PATT PATTERN FIFO HI

The two locations are on a LW boundary to allow 32 bit writes to the lower address with automatic transfer of the LSW and then the MSW. When the upper word is written the hardware automatically moves the data to the Pattern FIFO. The register values can be read-back if desired.

**IpParTtlPATT_Div**

x10 // 8

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Div |

FIGURE 11 IP-PARALLEL-TTL-PATT DIV BIT MAP

**Div** is programmed with the divisor used for the first level of division within the user specified counter to operate the Pattern Generator. The reference for the counter is the 100 MHz in this design.

100 MHz/[Div + 1] is the formula. Set Div = 4 to create 20 MHz reference

The Pattern Generator uses either 2 or 4 states to transmit each output state. 2 are used in the “User Pattern” type 0 and 4 for all internal patterns. Programming to 20 MHz provides a 5 MHz ramp update rate for example.

**IpParTtlPATT_PattFifoCnt**

x12 // 9 Command FIFO Count

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Pattern FIFO Count |

FIGURE 12 IP-PARALLEL-TTL-PATT PATTERN FIFO COUNT

Read this location to determine the number of patterns currently stored. xFFF is max count.

**IpParTtlPATT_CosFallingEn**

x18 // 12 COS Falling Enable

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | COS Falling Enable |

FIGURE 13 IP-PARALLEL-TTL-PATT COS FALLING ENABLE

The bits in the Falling Enable register are set to enable falling edge detections of the corresponding IO bit. When cleared the falling edges of the IO bit are ignored.

IO 47-32 are used for GPIO. Bit 0 in this register corresponds to IO(32).

**IpParTtlPATT_CosRisingEn**

x1A // 13 COS Rising Enable

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | COS Rising Enable |

FIGURE 14 IP-PARALLEL-TTL-PATT COS RISING ENABLE

The bits in the Falling Enable register are set to enable rising edge detections of the corresponding IO bit. When cleared the rising edges of the IO bit are ignored.

**IpParTtlPATT_CosEdgeLevel**

x1C // 14 Edge Level Select Control Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | EdgeLevel |

FIGURE 15 IP-PARALLEL-TTL-PATT EDGELEVEL BIT MAP

Each Bit of the EdgeLevel Control Register corresponds to a bit in the Parallel Port. When set the bit is selected to be COS processed with Edge. When cleared the bit is selected for Level processing. The type of processing coupled with the interrupt enable

selects how a particular IO is handled. The direct data is always available. The Rising and Falling status are always available. EdgeLevel is applied to filter which potential interrupters of each type are selected and the Interrupt Enable further selects which subset of those can actually generate an interrupt.

For example the direct data is filtered with EdgeLevel and only the level bits make it to the Filtered data [also treated with Polarity]. After the Filtered level the Interrupt Enable is applied to potentially allow interrupts from those sources.

**IpParTtlPATT_CosIntEn**

x1E // 15 Interrupt Enable Control Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Interrupt Enable |

FIGURE 16 IP-PARALLEL-TTL-PATT INTEN BIT MAP

Each bit of the IntEn registers corresponds to an input bit. When set the IO can cause an interrupt. When cleared that IO is masked out. Please note: EdgeLevel selects which type of pre-processing is applied to the Input prior to the interrupt enable. Polarity, Rising, Falling are used to select the active level or edge to process.

**IpParTtlPATT_CosPolarity**

x20 // 16 Polarity Control Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Polarity |

FIGURE 17 IP-PARALLEL-TTL-PATT POLARITY BIT MAP

Each Bit of the Polarity Control Register corresponds to a bit in the Parallel Port. When set the sense of the filtered data is inverted, when cleared the received data is left alone. For example, if the receive signal is active low the corresponding polarity bit is set to have it reflected as an active high signal in the filtered data. If the bit is to be used as an interrupt the enable is ANDed with the filtered data making the sense important to program. If not used for interrupt purposes it is up to the user to decide whether inversion is helpful or not. Please note: polarity does not affect the Direct Data port.

**IpParTtlPATT_GpioDir**

x22 // 17 Direction Control Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Direction |

FIGURE 18 IP-PARALLEL-TTL-PATT GPIO DIRECTION BIT MAP

Each Bit of the Direction Control Register corresponds to a bit in the Parallel Port. When set the IO is programmed to transmit, when cleared the IO is programmed to receive.

**IpParTtlPATT_GpioOut**

x24 // 18 Direct Output defininition

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Tx Data Definition |

FIGURE 19 IP-PARALLEL-TTL-PATT GPIO TX DATA

Data to transmit from the GPIO port is written to this register. The data is masked with the Direction register – only enabled bits are driven.

**IpParTtlPATT_GpioDirect**

x26 // 19 Direct IO Read

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | IO Definition |

FIGURE 20 IP-PARALLEL-TTL-PATT GPIO DIRECT DATA

The current state of the IO lines are returned from this register. Synchronizing to the IP clock is the only filtering done with this port.

**IpParTtlPATT_HalfDiv**

x28 // 20

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | HalfDiv |

FIGURE 21 IP-PARALLEL-TTL-PATT HALFDIV BIT MAP

**HalfDiv** is programmed with the divisor used for the first level of division within the user specified counter. The reference for the counter is the 100 MHz in this design. The programmed clock is used in the GPIO COS hardware.

The output of the user specified counter is further divided by 2 to create a square wave no matter the programmed divisor set.

Example: Desire 1 MHz. reference for Transmission. Last stage is /2 => divide 100 MHz to get 2 MHz. = 50. The first stage divider is preloaded with x1 and rolls over to x1 when the programmed count is reached.

**IpParTtlPATT_CosRisingSt**

x2A // 21 COS Rising Status

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | COS Rising Status |

FIGURE 22 IP-PARALLEL-TTL-PATT RISING STATUS

The bits in the Rising Status register are set when the selected IO transitions from low to high. The programmed sample rate can affect how long it takes for the transition to be reflected in the status. See HalfDiv for more information. Please note: the corresponding CosRisingEn bit will need to be set in order to enable the capture of the corresponding rising edge. Status is cleared by writing to this port with the status bit to clear set.

**IpParTtlPATT_CosFallingSt**

x2C // 22 COS Falling Status

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | COS Falling Status |

FIGURE 23 IP-PARALLEL-TTL-PATT FALLING STATUS

The bits in the Falling Status register are set when the selected IO transitions from high to low. The programmed sample rate can affect how long it takes for the transition to be reflected in the status. See HalfDiv for more information. Please note: the corresponding CosFallingEn bit will need to be set in order to enable the capture of the corresponding falling edge. Status is cleared by writing to this port with the status bit to clear set.

**IpParTtlPATT_CosFiltered**

x2E // 23 Filtered Data

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Filtered Data |

FIGURE 24 IP-PARALLEL-TTL-PATT FILTERED DATA

Read only port. The value of the port is the result of applying the EdgeLevel and Polarity to the IO. Please note: Direction is not a filter – transmitted bits can be filtered inputs if desired. The Polarity is applied on a bit-by-bit basis and then masked by the EdgeLevel definition to present the Level Selected data with the desired polarity. Please note: the output of the Filtered data is ANDed with the Interrupt Enable bits and then ORed together to create the level type interrupt request.

**IpParTtlPATT_Test0**

x30 // 24 Test port IO 15-0

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | IO15-0 |

FIGURE 25 IP-PARALLEL-TTL-PATT TEST DATA 0

R/W port with the read returning the state of the IO lines for the associated bit. When TestIoEn is set, the contents of this register are driven onto the IO lines to support BIT.

**IpParTtlPATT_Test1**

x32 // 25 Test port IO 31-16

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | IO31-16 |

FIGURE 26 IP-PARALLEL-TTL-PATT TEST DATA 1

R/W port with the read returning the state of the IO lines for the associated bit. When TestIoEn is set, the contents of this register are driven onto the IO lines to support BIT.

**IpParTtlPATT_StartLo**

x34 // 26 Start Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Lower ½ of Start value |

FIGURE 27 IP-PARALLEL-TTL-PATT START LO

**IpParTtlPATT_StartHi**

x36 // 27 Start Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Upper ½ of Start value |

FIGURE 28 IP-PARALLEL-TTL-PATT START HI

R/W port to set the Pattern Start Value. Generally, the first pattern output. Has various meanings depending on the Type selected. See Type section for more information. On LW boundary to allow 32 bit auto transfers.

**IpParTtlPATT_StopLo**

x38 // 28 Stop Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Lower ½ of Stop value |

FIGURE 29 IP-PARALLEL-TTL-PATT STOP LO

**IpParTtlPATT_StopHi**

x3A // 29 Stop Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Upper ½ of Stop value |

FIGURE 30 IP-PARALLEL-TTL-PATT STOP HI

R/W port to set the Pattern Start Value. Generally, the last pattern output. Has various meanings depending on the Type selected. See Type section for more information. On LW boundary to allow 32 bit auto transfers.

**IpParTtlPATT_CntLo**

x3C // 30 Count Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Lower ½ of Count value |

FIGURE 31 IP-PARALLEL-TTL-PATT COUNT LO

**IpParTtlPATT_CntHi**

x3E // 31 Count Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Upper ½ of Count value |

FIGURE 32 IP-PARALLEL-TTL-PATT COUNT HI

R/W port to set the Pattern Count Value. Usually the number of cycles to perform. 0x00 is used for continuous operation. A cycle is defined as the sequence through the programmed mode. For example, a Ramp starts with the Start value, increments with the slope, and stops with or near the Stop value. This sequence is one cycle. Has various meanings depending on the Type selected. See Type section for more information. On LW boundary to allow 32 bit auto transfers.

**lpParTtlPATT_RxFifoRdLo**

x40 // 32 Rx FIFO Data

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Lower ½ of RX FIFO |

FIGURE 33 IP-PARALLEL-TTL-PATT RX FIFO LO

**lpParTtlPATT_RxFifoRdHi**

x42 // 33 Rx FIFO Data

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Upper ½ of RX FIFO |

FIGURE 34 IP-PARALLEL-TTL-PATT RX FIFO HI

Read only port on LW boundary. Reading lower value “reads” FIFO. Access lower then upper. Set-up for 32 bit access to the 16 bit ports. Data is stored from loop-back or external capture operations.

**lpParTtlPATT_RxFifoCnt**

x44 // 34 Rx FIFO Count

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | RX FIFO Count |

FIGURE 35 IP-PARALLEL-TTL-PATT RX FIFO COUNT

Read only port with current count of data stored in the Rx Data FIFO. xFFF is max count.

**IpParTtlPATT_RxFifoAfl**

x46 // 35 Rx FIFO Almost Full

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | RX FIFO Almost Full |

FIGURE 36 IP-PARALLEL-TTL-PATT RX FIFO AFL SETTING

R/W port where user can specify the almost full FIFO condition. When the Rx Data FIFO count is greater than the value stored in this register the Almost Full signal is asserted. A latched version is stored in the Status Register and can be used for an interrupt request. Depending on the system operation the setting can be for the expected size of the data to receive. It can be set to make sure the FIFO does not over run – that there is enough time to read the FIFO etc. Depending on the expected reception frequency and amount of data the optimal level is set.

**IpParTtlPATT_Slope**

x48 // 36 Slope Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Pattern Slope Register |

FIGURE 37 IP-PARALLEL-TTL-PATT SLOPE REGISTER

The Slope Register sets the amount to increment or decrement per period for pattern generation. For a Rising Ramp it is the amount to add to the previous value. For a Falling Ramp it is the amount to subtract from the previous value. 16 bits allow up to 64K per period. The next address is skipped allowing the Slope to be expanded to a 32 bit value should the requirement arise.

**IpParTtlPATT_HCount**

x4C // 38 Horizontal Count Register

| DATA BIT | DESCRIPTION |
|---|---|
| 15-0 | Pattern Horizontal Count Register |

FIGURE 38 IP-PARALLEL-TTL-PATT HCOUNT REGISTER

The Horizontal Count Register sets the number of periods for horizontal values during a cycle. For example, with a trapezoidal waveform the structure is rising ramp, horizontal, falling ramp, horizontal. The rising ramp section will incorporate the final value, it may make sense to program the horizontal value to one less than the number of periods in the ramp. Start = 0, Stop = 20 Slope = 1 -- 21 periods to go from 0x00 to x14. Using 21 periods horizontal will result in 22 of the Stop value since the end point is the same point in this example.

16 bits allow up to 64K horizontal periods per section. It is possible to generate a really slow clock with fast, sloped edges if desired.

## Type Details

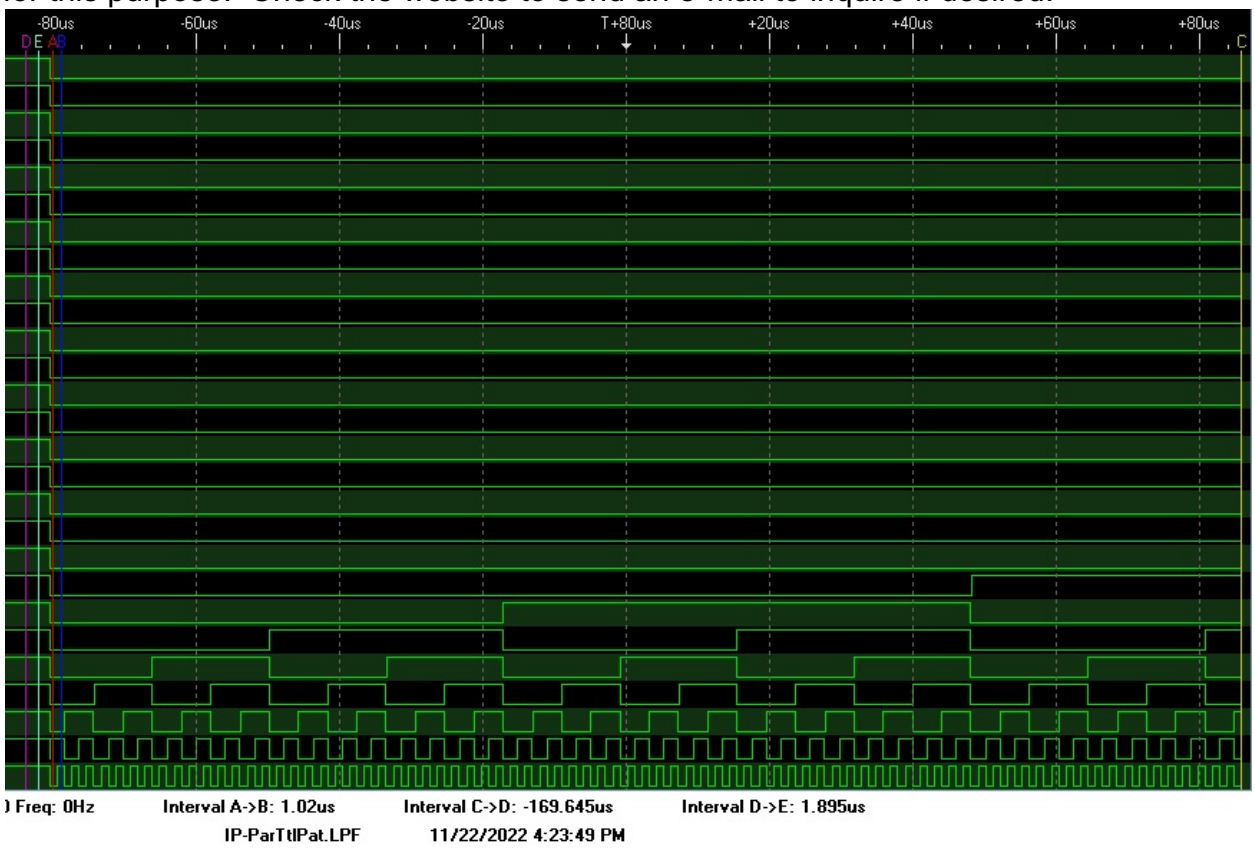
Start, Stop, and Count are 32 bit quantities and treated as unsigned integers. 0x00 = Min value, and xFFFF FFFF max value. The exception is a zero count decodes to run forever – stop with the enable.

Slope, HCount are 16 bit values and treated as unsigned integers.

Please refer to the Type descriptions to see how the parameters are used for a particular waveform.

**Type 0**

Select to broadcast a user developed pattern. Data is organized in the host memory and pushed to the Pattern FIFO. The Pattern FIFO is read by the Pattern state-machine and output onto the IO31-0 lines. The output is valid for 2 reference clock periods to allow the external clock to be generated with 50-50 timing. Experiments using the Win10/11 SW package show a 500 KHz+ limit for a continuous output in this mode. If the pattern to be broadcast completely fits within the FIFO, higher frequencies can be used. The data, once read from the FIFO is no longer available requiring repeating patterns to be resent. We are considering a Dual Port RAM implementation for this purpose. Check the website to send an e-mail to inquire if desired.




In this case the User Pattern is a Ramp. The initial value is zero. The update rate is 1 MHz.

**Type 1**

Select for Rising Ramps. The Start, Stop, Slope, and Count registers are used along with the programmed frequency to control the output. 4 clocks per output period. Start is the initial value broadcast. Slope is added repeatedly until the final value is reached. The Start value immediately follows. The Count is incremented for each set [cycle] and the process stopped when the terminal count reached.

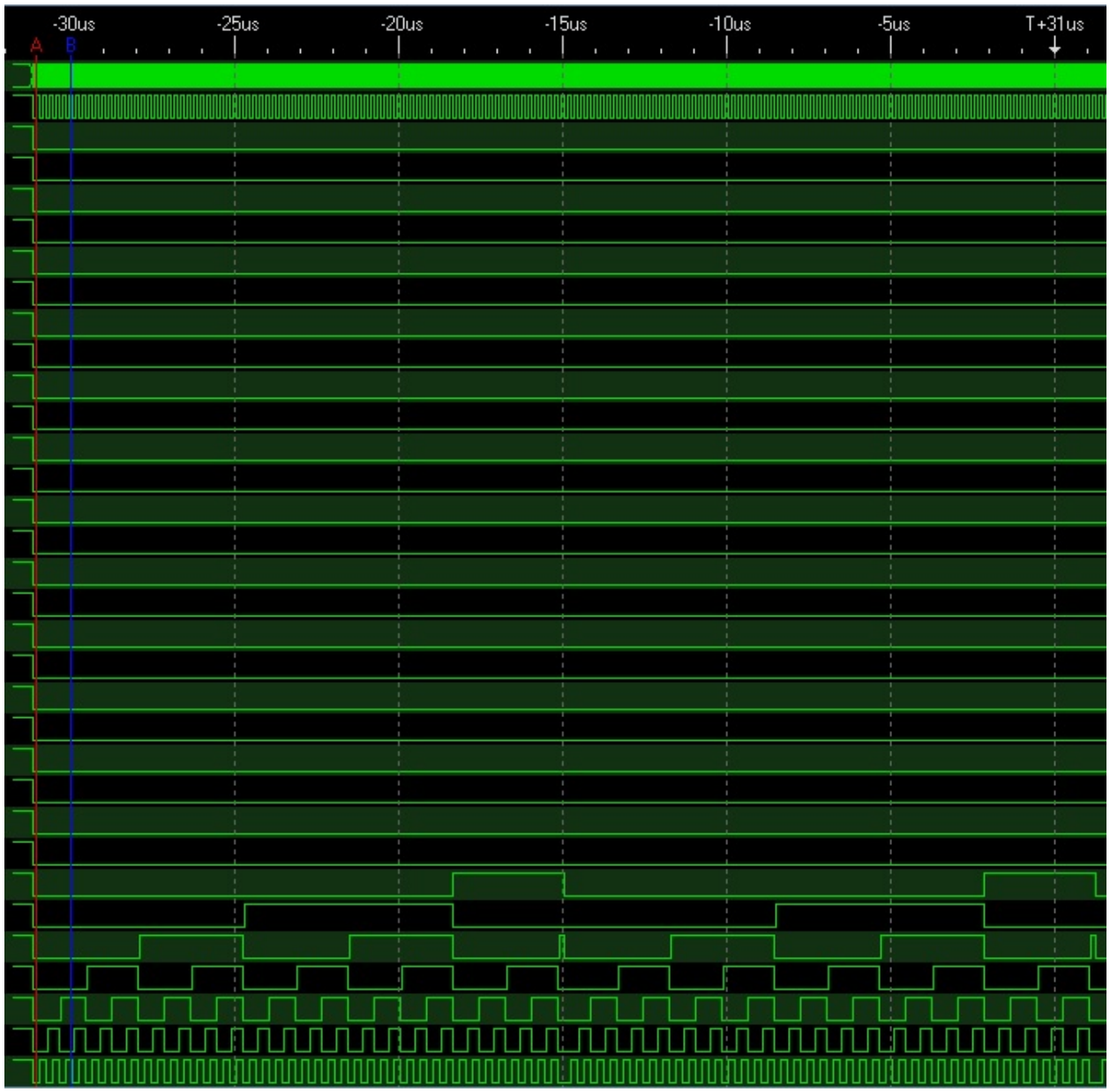
The end value will be the stop value if the slope divides evenly into the range set by the Start and Stop registers. If not an even multiple and the Stop value is not at the MaxCount [xFFFF FFFF] the pattern will over shoot by the programmed amount and return to the Start value from there. If the current value is within the MaxCount – Slope range the Ramp is returned to the Start Value.

Start = 0, Stop = x10, Slope = 3. 0, 3, 6, 9, xC, xF, x12, 0… results since the xF value is less than the end point and not near the boundary.

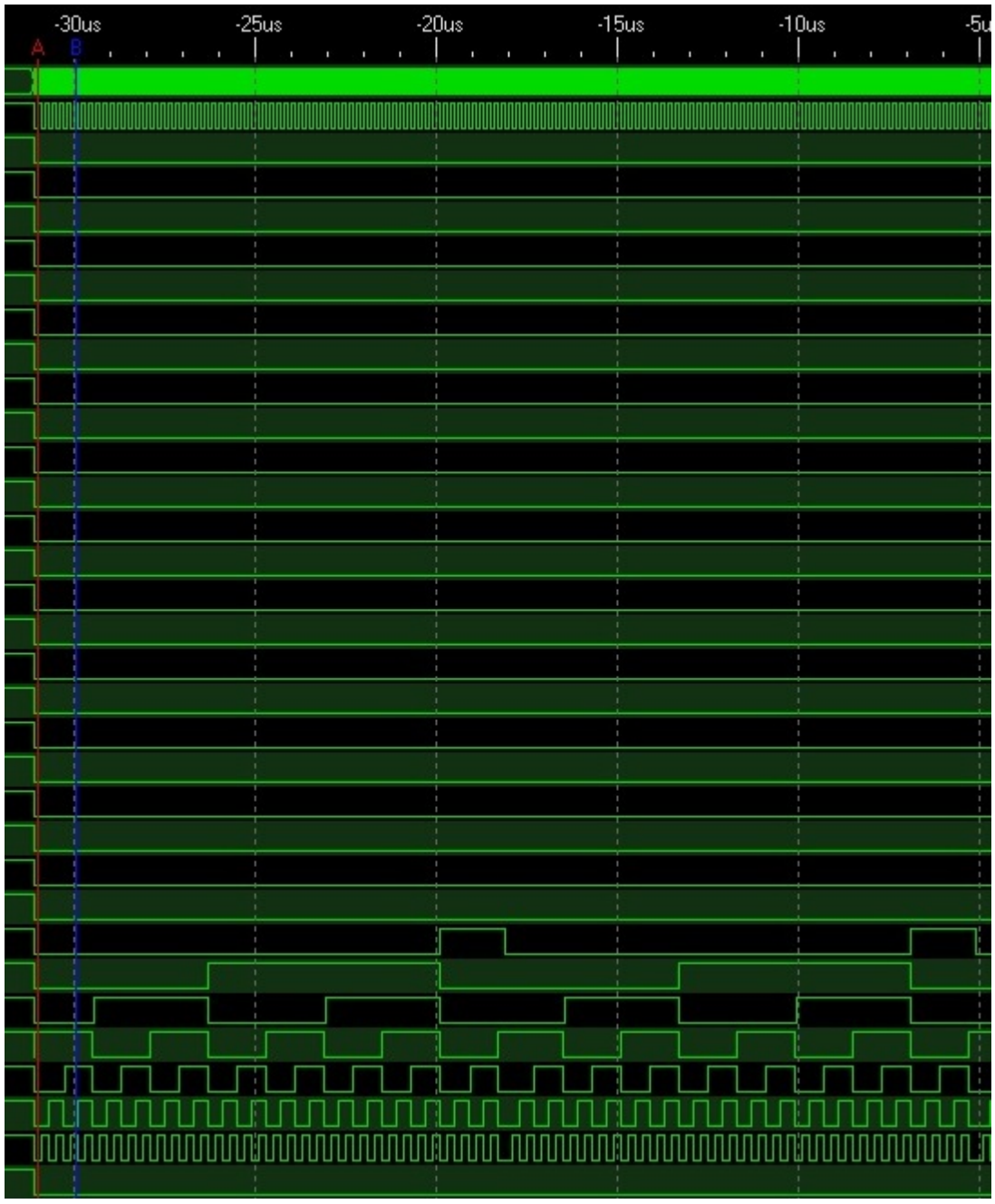
Start = 0, Stop = x10, Slope = 1 results in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, x10, 0 since the Slope divides evenly.

Notice both Start and Stop are inclusive leading to (Stop – Start)/Slope + 1 for the number of steps when an even divisor is used.

The reference logic analyzer diagram counts from x00 to x50. Slope is x1. The terminal value is one count [x50] and the next value is 0x00. First two of Count = 5 are shown.



Rising Ramp.  x00 -x50 Slope = 1

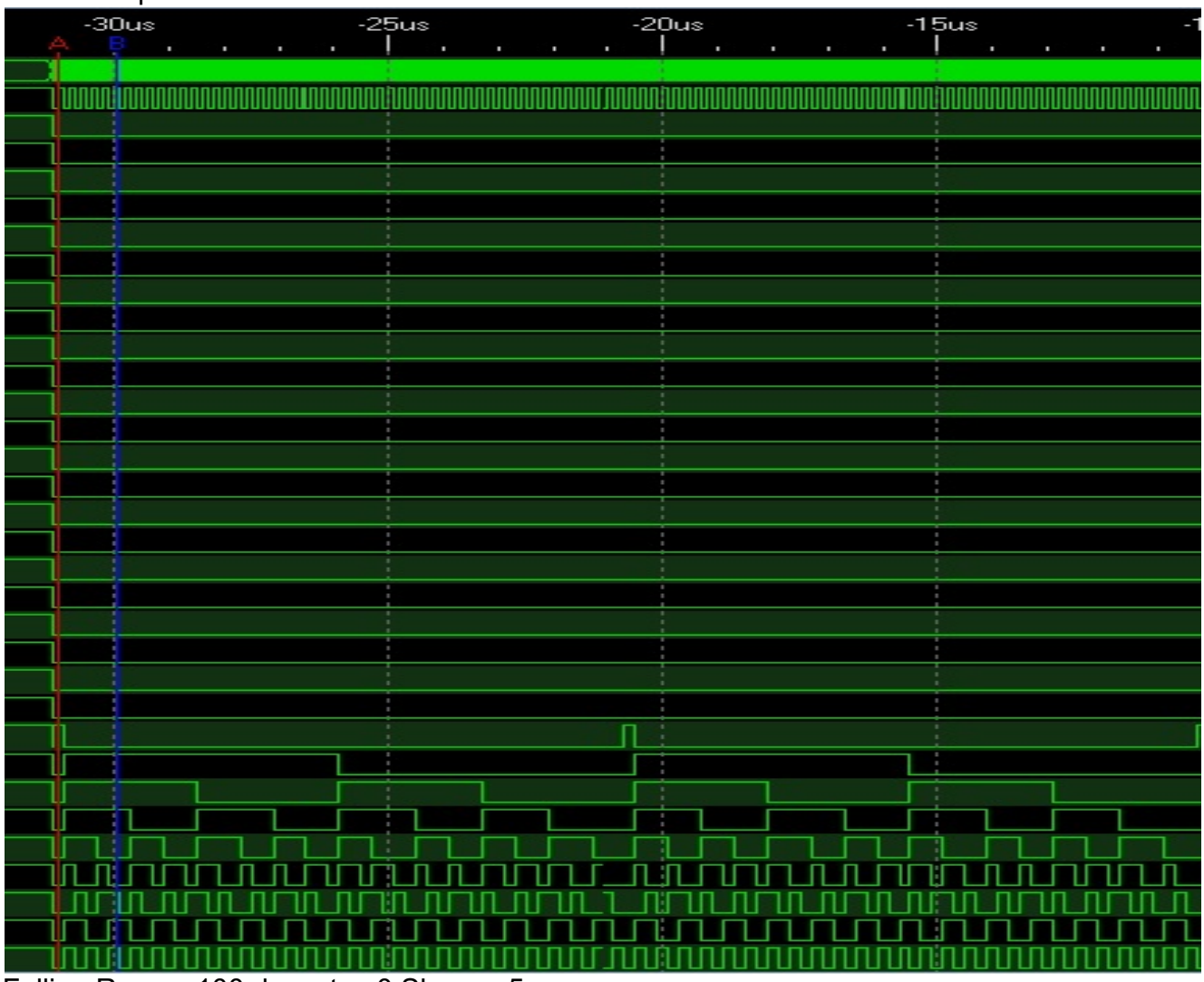


Rising Ramp.  x10 -x90 Slope = 2

**Type 2**

Select for Falling or Decreasing Ramps. Very similar to Type 1 except the Start Value is the upper value and the Stop value is the lower value. The Slope is subtracted each period until <= Stop value with a test to make sure not going <0.

Start = x20, Stop = x10 Slope = 3. x20, x1D, x1A, x17, x14, x11, xE, x20 is the resulting pattern. Since x11 is > Stop and the result is not negative the undershoot is performed. Even multiples eliminate the undershoot.

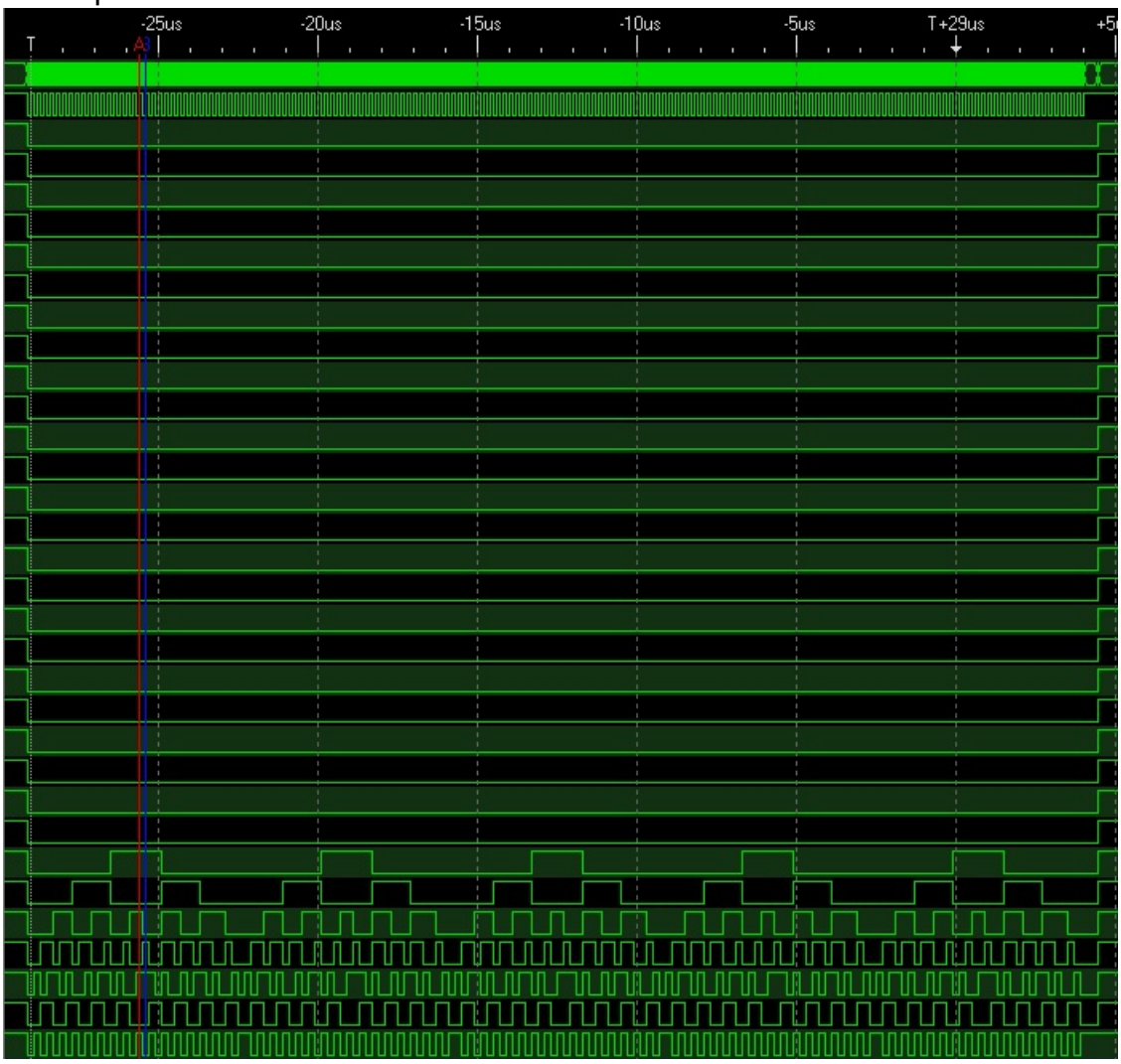



Falling Ramp x100 down to x0 Slope = 5

**Type 3**

Combine Rising and Falling ramps to create a Pyramid shape. The Rising section works as described above for the first ramp. At the termination of the Rising Ramp the initial value for the Falling Ramp – slope is used to keep the “peak” to a single value. Note the overrun checks and boundary behavior apply. The first value of the falling ramp is based on the Stop and Stope values. Similarly, at the transition from Falling to Rising the first value is the Start Value plus the slope. This means the waveform can have over and under shoot as desired and stay on track.

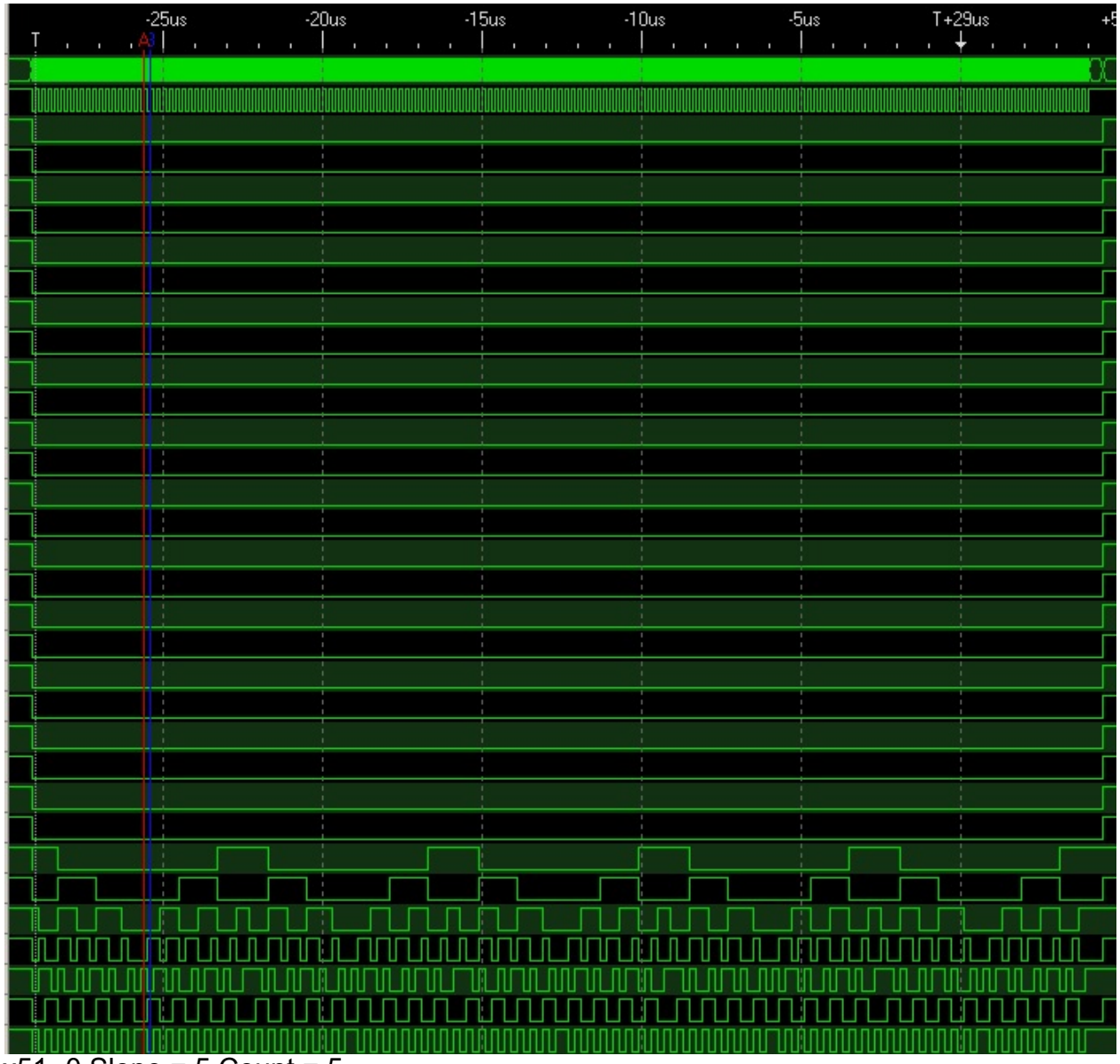
Start = 0, Stop = x10, Slope = 1 results in 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, x10, F, E, D, C, B, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0. Since Slope divides into the ramps, the end points are the termination points. The combination up and down is counted as 1 for the Count parameter.



The reference logic analyzer photo demonstrates x0 – x51 with a slope of 5 and a count of 5. Note: after the transfer is complete the IO are released to the off state which is xFF for this board. The transition is visible on both sides of the wave form. The terminal count is x55 because the slope of 5 brings the rising side to x50 < x51. The last slope addition brings the output to x55 as a result. The next value is x51-x5 = x4C.

**Type 4**

Combine Falling and Rising ramps to create an inverted Pyramid shape. The Falliing section works as described above for the Type 2 ramp. At the termination of the Falling Ramp the initial value [Stop] for the Rising Ramp + slope is used to keep the “peak” to a single value. Note the overrun checks and boundary behavior apply. The first value of the second Falling ramp is based on the Start and Stope values. Similarly, at the transition from Falling to Rising the first value is the Start Value plus the slope. This means the waveform can have over and under shoot as desired and stay on track.




x51- 0 Slope = 5 Count = 5

The logic analyzer screen shot shows the repeated Inverted Pyramid shape. Since the initial count is x51 the next to last value minus the slope would be negative. The design uses the last non-negative x1 value as the last value on the falling ramp. The initial value on the rising portion is based on 0x00 due to the programmed stop value. For a x1, x5 sequence at the valley. At the peak the max value is allowed to over-run since it would not wrap around in that case.

Another example:
Start = x10, Stop = x0, Slope = 1 results in x10, F, E, D, C, B, A, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, x10 Since Slope divides into the ramps, the end points are the termination points. The combination down and up is counted as 1 for the Count parameter.

**Type 5**

Select for a Square wave output. Start, Stop, Count values are used to control this type. Start is output followed by Stop. 2 of the programmed clock periods per side to keep consistent with the other modes. Start, Stop = one count for the count parameter. It is permissible to have Stop > Start and vice-versa to allow either start state as well as offsets for the switching.

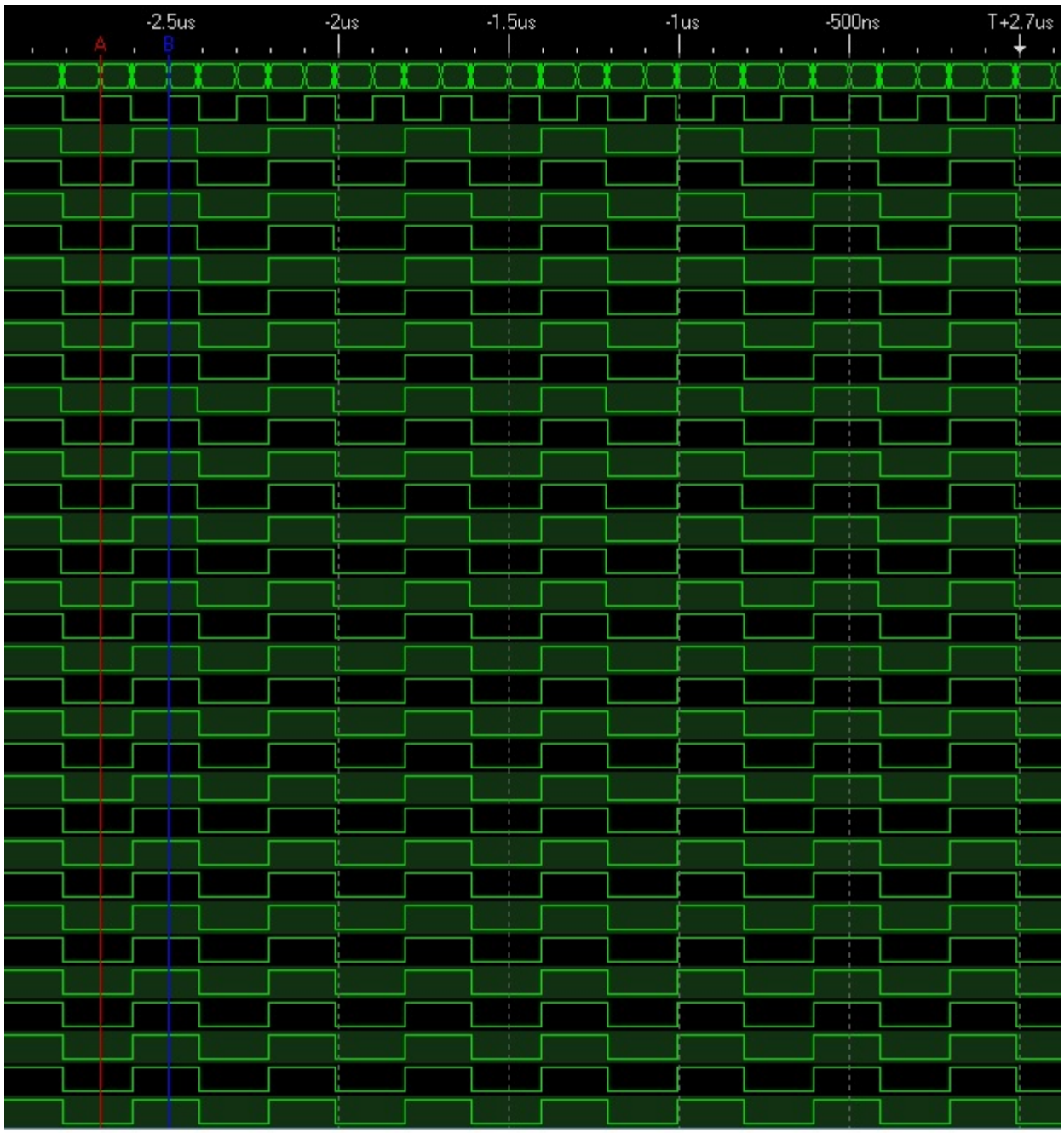
Please note: up to 32 “clocks” can be generated with 0x00 xFFFF FFFF for the start Stop values. It is recommended to have non-utilized outputs set to the same value for both Start and Stop to reduce switching noise and power consumption. Since using the IO as individual bits in this case the value is really going from 0 – 1.

Examples: 0x0000 0000 xFFFF FFFF will produce 32 clock outputs or a switching pattern with a clock reference [GPIO top bit].

0xFFFF FF00 0xFFFF FFFF will have 8 clocks switching and 24 held high throughout.

0xAAAA AAAA 0x5555 5555 will have 16 clocks with rising edge and 16 with falling edge activity. If you don’t need the phases aligned this programming will be closer to neutral from a power / EMI perspective.

In the logic analyzer screen shot is an example of the min-max combination of 0x00 and 0xFFFF FFFF for the Start and Stop values. The upper trace is the reference clock. The A-B timing is 200 nS in this example.

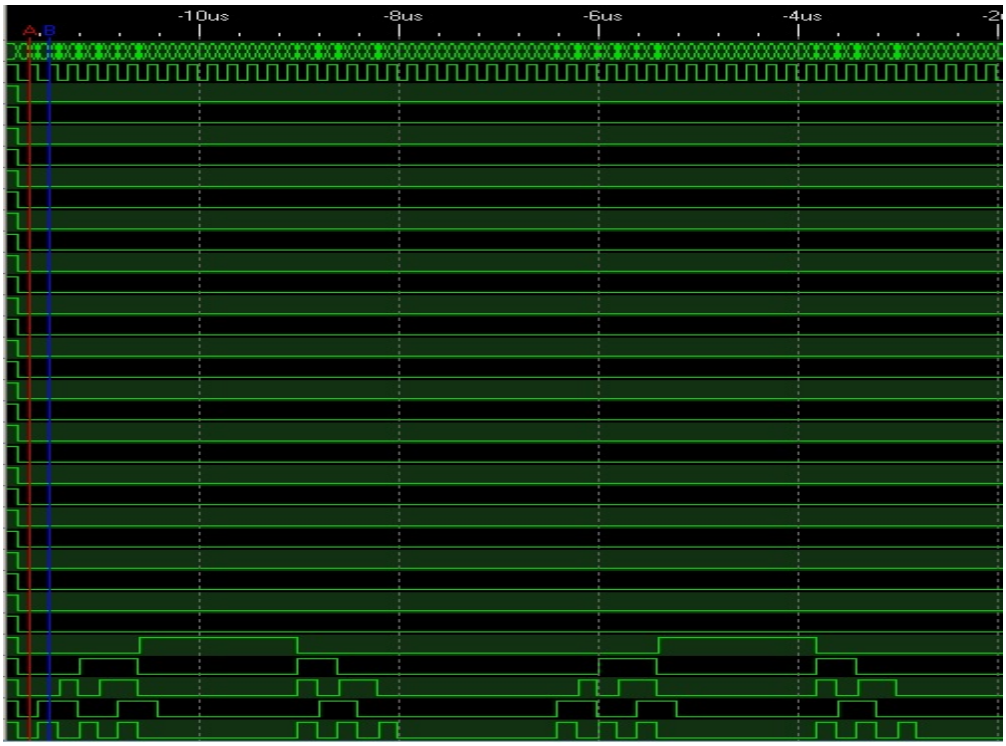



Start=0x0000 0000  Stop=xFFFF FFFF

**Type 6**

Trapezoidal waveforms introduce horizontal timing. Ramp Up, Horizontal, Ramp Down, Horizontal. This type uses Start, Stop, Slope, HCount, and Count to control the operation. Ramp Up uses Start, Stop and slope to create the first edge. Once at the Stop or overshoot value, HCount is used to add the horizontal component [at the Stop value]. Each HCount represents 4 of the reference clocks [same as the rising and falling sections]. At the end of the HCount, the falling ramp is initiated with the Stop-Slope value. The falling ramp is executed. After the falling ramp a second HCount section is generated with the Start value output. With the programmable over-run and under-run values various simulations can be created based on the Slope and steps between Start and Stop. The Count is based on the combination of Rising, Horizontal, Falling, Horizontal.

Example:
Start = x10 Sto p= x20 Slope = 3 HCount = 4
x10, x13, x16, x19, x1c, x1f, x22, x20, x20, x20, x20, x1d, x1a, x17, x14, x11, xe, x10, x10, x10, x10.



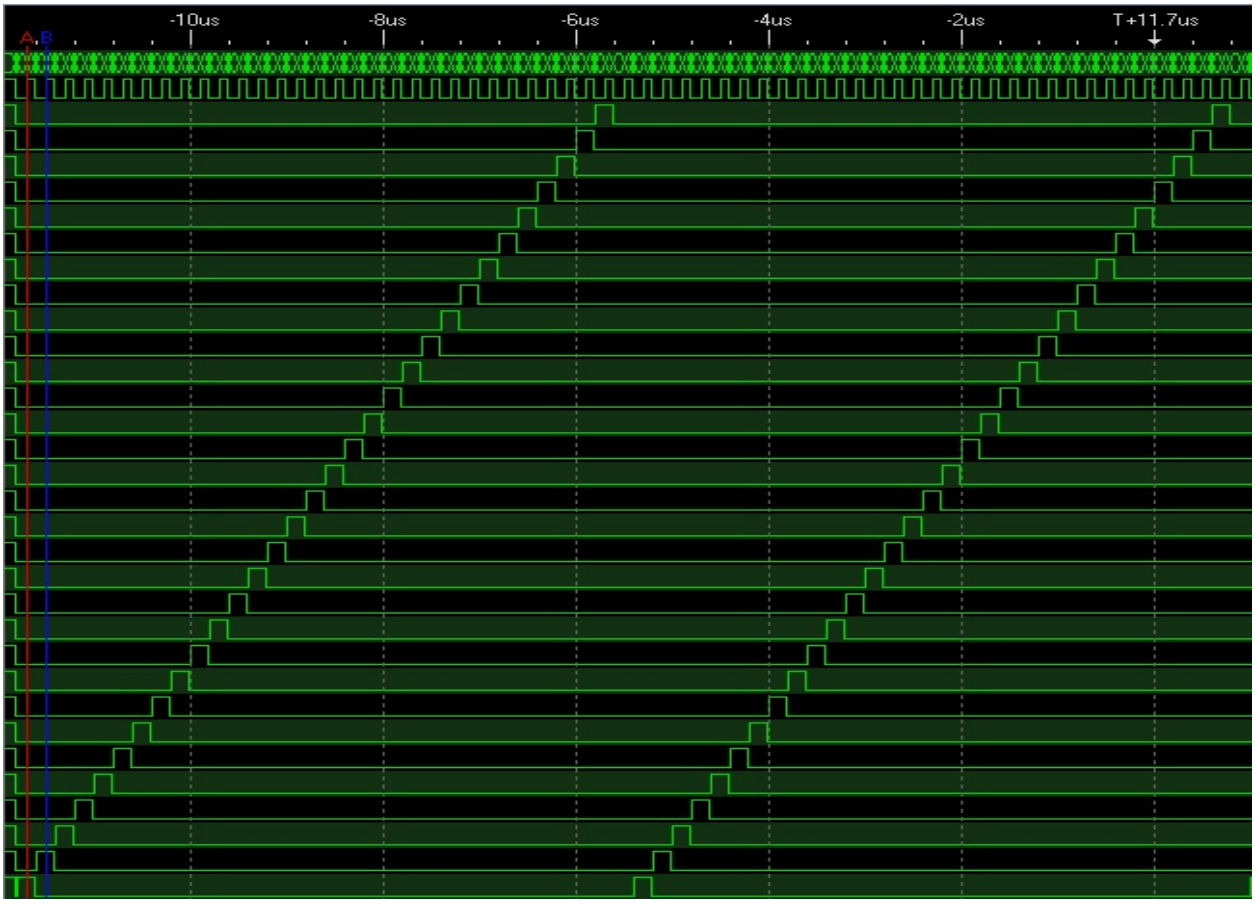
Trapezoidal Start = x0 Stop = x10 Slope = x3 HCount = 7

**Type 7**

Shift Up acts like a shift register where the Start Value is shifted up until Greater- equal to the Stop Value. This is one count. Slope, and HCount are not used for this type.

Example: Start = 0x0000 0001 Stop = x8000 0000. The output will double with each shift: x1, x2, x4… x8000 0000. 32 shifts for the one cycle. Alternate values can be used for example Start = x3 will produce x3, x6, xC, x18. This pattern provides a quick method to determine all of the IO are independent and functioning.

This Shift Up pattern is programmed for 20 MHz reference yielding a 5 MHz pattern. Start value of 0x0000 0001.



## Interrupts

IP-Parallel-TTL-PATT interrupts are treated as auto-vectored on many carriers. When the software enters into an exception handler to deal with an IP-Parallel-TTL-PATT interrupt the software must read the status register(s) to determine the cause(s) of the interrupt, clear the interrupt request(s) and process accordingly. Power on initialization will provide a cleared interrupt request and interrupts disabled.

The interrupt is mapped to INT0 on the IP connector, which is mapped to a system interrupt via the host [carrier] device. The source of the interrupt is obtained by reading the Interrupt Status registers. The status remains valid until that bit in the status register is cleared.

When an interrupt occurs, the Master interrupt enable should be cleared and the status register read to determine the cause of the interrupt. Next perform any processing needed to remove the interrupting condition, clear the status and enable the interrupt again.

The individual enables operate after the interrupt holding latches, which store the interrupt conditions for the CPU. This allows for operating in polled mode simply by monitoring the Interrupt Status register.

The base level has a master interrupt enable that affects all interrupt sources.

An interrupt Vector can be read for systems requiring vectored interrupt processing.

The Windows and Linux SW packages provide both straight Interrupt Status read and ISR status read. The ISR when activated by an interrupt request, disables the master interrupt enable and reads the Rising, Falling, and Filtered data. The Rising and Falling status is then cleared within the handler. Reading the ISR structure returns all of the interrupt status saving multiple application level calls. The standard Interrupt Status read only returns the status without any other operations.

## Loop-back

The Engineering kit has reference software, which includes an external loop-back test. The test requires external connections. The Acceptance Test Procedure [ATP] for this device uses the internal tests without a fixture attached. The external tests make use of the separated Data In and Data Out buses. The Rx function is enabled and pattern data captured.

We use IP-Debug-IO and IP-Debug-Bus to allow connection to the carrier and access to the testpoints on IO signals. For development a logic analyzer was attached to the header connections on IP-Debug-IO. All timing was verified in this manner.

The logic analyzer photos used to describe the Type programming were captured running the ATP SW and using IP-Debug-Bus and IP-Debug-IO along with the logic port connections.

## ID PROM

Every IP contains an ID PROM, whose size is at least 32 x 8 bits. The ID PROM aids in software auto configuration and configuration management. The user's software, or a supplied driver, may verify that the device it expects is actually installed at the location it expects, and is nominally functional. The ID PROM contains the manufacturing revision level of the IP. If a driver requires that a particular revision be present, it may check for it directly. The revision is also readable from the base revision register where both the major [reported in the PROM] and minor fields are available.

The location of the ID PROM in the host's address space is dependent on which carrier is used.

Standard data in the ID PROM on the IP-Parallel-TTL-PATT is shown in the figure below. For more information on IP ID PROM's refer to the IP Module Logic Interface Specification.

| Address | Data | |
|---|---|---|
| 01 | ASCII "I" | ($49) |
| 03 | ASCII "P" | ($50) |
| 05 | ASCII "A" | ($41) |
| 07 | ASCII "H" | ($48) |
| 09 | Manufacturer ID | ($1E) Dynamic Engineering |
| 0B | Model Number | ($03) IP-Parallel-TTL |
| 0D | Revision | ($01) RevisionMajor |
| 0F | reserved | ($00) Customer Number |
| 11 | Driver ID, low byte | ($0D) Design Number-PATT |
| 13 | Driver ID, high byte | ($00) |
| 15 | No of extra bytes used | ($0C) |
| 17 | CRC | ($B1) |

FIGURE 39 IP-PARALLEL-TTL-PATT ID PROM

## IP-Parallel-TTL-PATT Logic Interface Pin Assignment

The figure below gives the pin assignments for the IP Module Logic Interface on IP-Parallel-TTL-PATT. Pins marked n/c below are defined by the specification, but not used on the IP-Parallel-TTL-PATT. Also see the User Manual for your carrier board for more information.

| | | | |
|---|---|---|---|
| GND | GND | 1 | 26 |
| CLK | +5V | 2 | 27 |
| Reset* | R/W* | 3 | 28 |
| D0 | IDSEL* | 4 | 29 |
| D1 | n/c | 5 | 30 |
| D2 | MEMSEL* | 6 | 31 |
| D3 | n/c | 7 | 32 |
| D4 | INTSEL* | 8 | 33 |
| D5 | n/c | 9 | 34 |
| D6 | IOSEL* | 10 | 35 |
| D7 | n/c | 11 | 36 |
| D8 | A1 | 12 | 37 |
| D9 | n/c | 13 | 38 |
| D10 | A2 | 14 | 39 |
| D11 | n/c | 15 | 40 |
| D12 | A3 | 16 | 41 |
| D13 | INTREG0* | 17 | 42 |
| D14 | A4 | 18 | 43 |
| D15 | n/c | 19 | 44 |
| BS0* | A5 | 20 | 45 |
| BS1* | n/c | 21 | 46 |
| n/c | A6 | 22 | 47 |
| n/c | Ack* | 23 | 48 |
| +5V | n/c | 24 | 49 |
| GND | GND | 25 | 50 |

NOTE 1: The no-connect signals above are defined by the IP Module Logic Interface Specification, but not used by this IP. See the Specification for more information.

NOTE 2: The layout of the pin numbers in this table corresponds to the physical placement of pins on the IP connector. This table may be used to easily locate the physical pin corresponding to a desired signal. Pin 1 is marked on the IP Module.

FIGURE 40 IP-PARALLEL-TTL-PATT LOGIC INTERFACE

## IP-Parallel-TTL-PATT IO Pin Assignment

The figure below gives the pin assignments for the IP Module IO Interface on IP-Parallel-TTL-PATT. Also see the User Manual for your carrier board for more information. IO#(signal name). Schematic shows IO#. Pairs are defined to be compatible with IP Standard Differential wiring – used on PCIe3IP, PCIe5IP etc.

| | | | |
|---|---|---|---|
| IO0 | IO24 | 1 | 26 |
| IO1 | IO25 | 2 | 27 |
| IO2 | IO26 | 3 | 28 |
| IO3 | IO27 | 4 | 29 |
| IO4 | IO28 | 5 | 30 |
| IO5 | IO29 | 6 | 31 |
| IO6 | IO30 | 7 | 32 |
| IO7 | IO31 | 8 | 33 |
| IO8 | IO32 | 9 | 34 |
| IO9 | IO33 | 10 | 35 |
| IO10 | IO34 | 11 | 36 |
| IO11 | IO35 | 12 | 37 |
| IO12 | IO36 | 13 | 38 |
| IO13 | IO37 | 14 | 39 |
| IO14 | IO38 | 15 | 40 |
| IO15 | IO39 | 16 | 41 |
| IO16 | IO40 | 17 | 42 |
| IO17 | IO41 | 18 | 43 |
| IO18 | IO42 | 19 | 44 |
| IO19 | IO43 | 20 | 45 |
| IO20 | IO44 | 21 | 46 |
| IO21 | IO45 | 22 | 47 |
| IO22 | IO46 | 23 | 48 |
| IO23 | IO47 | 24 | 49 |
| IO_GND | IO_GND | 25 | 50 |

NOTE: The layout of the pin numbers in this table corresponds to the physical placement of pins on the IP connector. This table may be used to easily locate the physical pin corresponding to a desired signal. Pin 1 is marked on the IP Module.

FIGURE 41 IP-PARALLEL-TTL-PATT CONNECTOR PINOUT

**Notes:**
IO_GND = DC ground for revision 04 PCB.
Pattern Generator(31-0) = IO 31-0
GPIO(15-0) = IO(47-32)

Pattern External Clock = IO 47
Pattern External Trigger = IO 46

External clock output with Mode Register selection otherwise GPIO controlled.
External clock input on same pin. Set to output to do Rx of transmitted pattern
External Trigger inputs are in parallel with COS operation.

# Applications Guide

## Interfacing

The pin-out tables are displayed with the pins in the same relative order as the actual connectors. Ideally the interconnecting cable is shielded and if very long it is recommended to separate the Address, Data, Control signals into separate groups to reduce switch induced cross talk.

**We provide the components. You provide the system**. Safety and reliability can be achieved only by careful planning and practice. Inputs can be damaged by static discharge, or by applying voltage outside of the TTL devices rated voltages. TTL signals have small series resistors located near the connector to provide some protection.

If induced noise is causing errors, please check the cabling and make sure the shields are properly tied to ground on one side. It may be necessary to go to higher grade cable.

## Construction and Reliability

IP Modules are conceived and engineered for rugged industrial environments. IP-Parallel-TTL-PATT is constructed out of 0.062 inch thick high temp ROHS compliant FR4 material.

Options are available for ROHS and standard processing

Through hole and surface mounting of components are used.

The IP Module connectors are keyed and shrouded with Gold plated pins on both plugs and receptacles. They are rated at 1 Amp per pin, 200 insertion cycles minimum. These connectors make consistent, correct insertion easy and reliable.

The IP is secured against the carrier with four metric M2 stainless steel screws. The heads of the screws are countersunk into the IP. The four screws provide significant protection against shock, vibration, and incomplete insertion. For most applications they are not required.

The IP Module provides a low temperature coefficient of .89 W/$^{o}$C for uniform heat. This is based upon the temperature coefficient of the base FR4 material of 0.31 W/m-$^{o}$C, and taking into account the thickness and area of the IP. The coefficient means that if .89 Watts are applied uniformly on the component side, then the temperature difference between the component side and solder side is one degree Celsius.

## Thermal Considerations

The IP-Parallel-TTL-PATT design consists of CMOS circuits. The power dissipation due to internal circuitry is very low. It is possible to create a higher power dissipation with the externally connected logic. If more than one Watt is required to be dissipated due to external loading forced air cooling is recommended. With one-degree differential temperature to the solder side of the board external cooling is easily accomplished.

## Warranty and Repair

Please refer to the warranty page on our website for the current warranty offered and options.

https://www.dyneng.com/warranty.html

## Service Policy

Before returning a product for repair, verify as well as possible that the suspected unit is at fault. Then call the Customer Service Department for a RETURN MATERIAL AUTHORIZATION (RMA) number. Carefully package the unit, in the original shipping carton if this is available, and ship prepaid and insured with the RMA number clearly written on the outside of the package. Include a return address and the telephone number of a technical contact. For out-of-warranty repairs, a purchase order for repair charges must accompany the return. Dynamic Engineering will not be responsible for damages due to improper packaging of returned items. For service on Dynamic Engineering Products not purchased directly from Dynamic Engineering, contact your reseller. Products returned to Dynamic Engineering for repair by other than the original customer will be treated as out-of-warranty.

## Out of Warranty Repairs

Out of warranty repairs will be billed on a material and labor basis. Customer approval will be obtained before repairing any item if the repair charges will exceed one half of the quantity one list price for that unit. Return transportation and insurance will be billed as part of the repair and is in addition to the minimum charge.

## For Service Contact:

Customer Service Department
Dynamic Engineering
150 DuBois St Suite B/C
Santa Cruz, CA 95060
831-457-8891
support@dyneng.com

## Specifications

| | |
|---|---|
| Host Interface: | IP Module 8 and 32 MHz capable |
| IO Interface: | 48 bit parallel port with 32 allocated to Pattern Generation interface and 16 as GPIO with COS. |
| COS sample rates generated: | 100 MHz for COS and Pattern Generation reference with SW Selectable divisors. |
| Software Interface: | Control Registers, Status Ports |
| Initialization: | Hardware Reset forces all registers [except vector] to 0. |
| Access Modes: | IO, Memory, ID, INT spaces (see memory map). Memory space is not utilized on this implementation. |
| Wait States: | minimized based on programmed clock rate |
| Interrupt: | Programmable per bit/function |
| Onboard Options: | Most Options are Software Programmable. Shunt for IO ground reference: open, DC, AC |
| Interface Options: | 48 IO plus reference on P2. |
| Dimensions: | Type II |
| Construction: | High temp ROHS compatible FR4 Multi-Layer Printed Circuit, Through Hole and SMT. |
| Temperature Coefficient: | .89 W/$^{O}$C for uniform heat across IP |
| Power: | Typical **180** mA @ 5V typical. |
| Temperature Range | –40C⇔85C or better rated components. Conformal Coating option for condensing environments |

## Order Information

| | |
|---|---|
| IP-Parallel-TTL-PATT | IP Module with 48 TTL IO. Programmable GPIO function with COS or Level based interrupts and status, Direct and Filterered data. Bit level Direction, Polarity, Rising and Falling control. 32 bit Pattern Generation with User input and 8 selectable patterns with User controls for limits, slope, quantity etc. |
| -CC | Conformal Coating option |
| -ROHS | Change to ROHS processing. Without this option, standard leaded solder will be used. |
| Eng Kit–IP-Parallel-TTL-PATT | IP-Debug-IO - IO connector breakout<br>IP-Debug-Bus - IP Bus interface extender<br>IP-Parallel-TTL-PATT Driver and reference software |

**Note**: The Engineering Kit is strongly recommended for first time purchases.