Chapter 8
Development of APF controller Software for DSP Implementation

Embedded System design involves choosing a right platform and the development support available from the vendor. The selection of the platform depends on processor, SOC, memory, Buses, software language, OS, code generating tools, simulators, emulators etc. Since the decision on DSP processor is made, the chapter covers the hardware and software development support tools for the development of system hardware and software.

DSK/EVM boards available from TI are chosen as the target system. The target system parallel connects the simulator in the host computer through a Target Server tool with board through parallel or serial lines from the host computer. *Emulator* provides a great flexibility and ease for developing various applications on a single system in place of testing multiple targeted systems. The Emulator uses the circuit consisting of the DSP processor. It emulates the target system with extended memory and with codes downloading ability during the edit-test-debug cycles. The hardware and software support tools used in the project are also described.

Active power filter for power of 75 kVA was build and tested [1]. The compensation process is instantaneous and control algorithm of the proposed APF based on the control strategy following from the instantaneous reactive power theory (Fig 8.1) was implemented in the fixed-point digital signal processor TMS320C50.

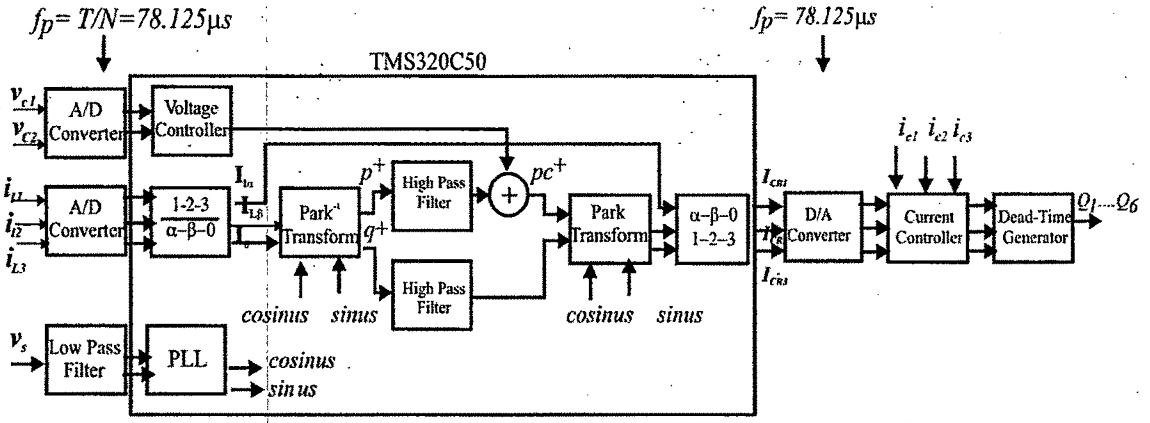


Fig 8.1 Block Diagram of APF Control Algorithm

The DSP implementation was used for single phase APF [3]. The proposed scheme [3] employs a control that requires less feedback information compared **to** other reported solutions. Only one current sensor is used to sense the nonlinear load current and two voltage sensors to sense the input supply voltage and the dc bus voltage. It also reduces the total chip count of the system using on-chip power electronics peripherals of the **DSP** controller **(TMS320F240).** The proposed method provides both harmonic elimination and power factor correction. The proposed scheme also requires fewer current sensors than other solutions. The technique requires only the load current, dc bus voltage and supply voltage information to estimate the fundamental current and hence calculate the required injected current using the proposed DSP algorithm.

(1 111 [illegible] 112) [illegible]

In our project we have used TMS320C6711 based DSP processor for implementation. The said series of processors are suitable for development of applications in the field of communication. Really we should have used TMS28XX series of DSP processors. Since the ccs link for 2000 series was not available and we wanted to use SIMULINK model and embedded target support by MATHWORKS for the development of embedded APF controller, the available tools for TMS320C67xx, daughter cards and software plug Ins are used. Brief summary of hardware and software development tools is given. The web sites of the manufacturer may be referred for further details.

8.1 Hardware Tools

Hardware tools used in the project for the development of embedded APF controller software includes:

- TMS 320C671X DSP Starter Kit
- ADS 8364 EVM (6 – input Analog to Digital Converter)
- TLV 2553 EVM (11 – input Analog to Digital Converter)
- TLV 5614 EVM (4 – output Digital to Analog Converter)
- 5 – 6 K Interface Card

8.1.1 TMS 320C671X DSK

The C671X DSP starter kit provides system design engineers with an easy-to-use, cost-effective way to take their high-performance TMS320C6000 designs from concept to production. It provides facility for fast development of networking, communications, imaging and other applications. Fig 8.2 depicts the DSK board. DSK operates at 150 MHz delivers an impressive 1200 MIPS and 600 MFLOPs. The C6711 DSK is a superset of the C6211 DSK. The kit includes a DSK board with ‘C6711/13 DSP processor, 16 MB external SDRAM, 128 KB External Flash for program and data storage, TI'S TLC320AD535 16-bit Data Converter, TI'Ss TPS56100 Power Management Device , JTAG Controller for emulation, Expansion Daughter Card Interface and CE-Compliant Universal Power Supply DSK

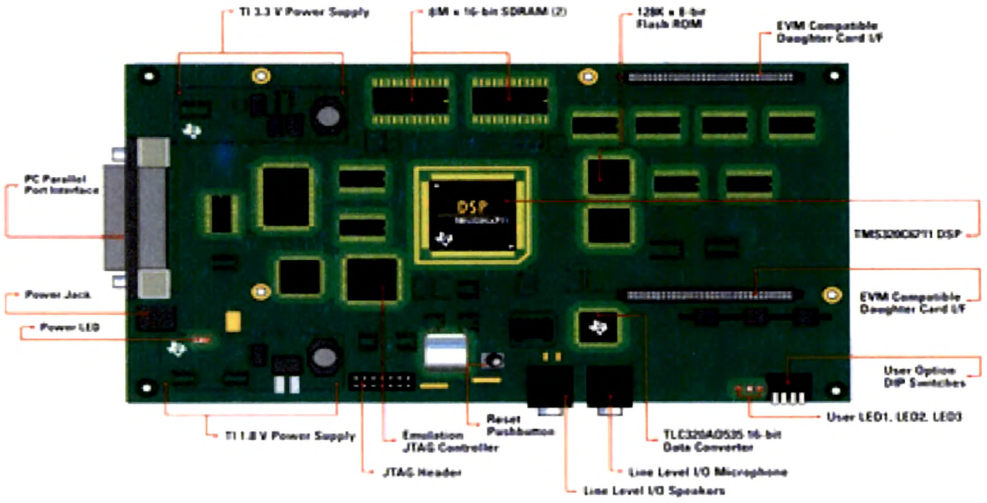


FIG. 8.2 TMS320C6711 DSK BOARD

8.1.2 Daughter Cards

Daughter Cards are hardware plug-on modules that compliment TI and/or Third Party DSP Starter Kits (DSKs) and development boards Daughter cards enable the evaluation of a wide range of data converters, prototype boards, new I/O interfaces and many other peripheral devices. They can be connected with TI DSP Starter Kits (DSKs) seamlessly via common connectors An introduction on the daughterboard concept is provided identifying the advantages of using this approach for rapid prototyping and accelerated product developments.

EVM daughter boards present a modular approach to interface design that can support a wide range of applications. Design efforts only need to focus on new interface requirements and related support software, rather than the complete DSP processing system. This incremental design paradigm lends itself well to rapid prototyping and product development acceleration. The different daughter cards are for:

- External Memory Interface
- Peripheral interface
- Host port interface

The EVM's two expansion connectors provide access to the DSP asynchronous external memory interface (EMIF) and its on-chip peripheral and control/status signals. Both connectors also provide multiple voltages to supply power to the daughter board. Each of the expansion connectors is a low-profile, 80-pin, 0.050"-pitch connector designed to support high-speed board interconnections. Mating connectors for the daughterboard are available in several mating heights which all meet the maximum height requirement defined in the PCI Local Bus Specification. Two standard sizes of EVM daughter boards are defined. A small size is intended for applications that do not require an I/O connection on the mounting bracket. A large size is intended for applications that require an I/O connection on the mounting bracket or need more board space for implementation. Fig 8.3 shows the daughterboard interface including DSP signals provided by the expansion connectors.

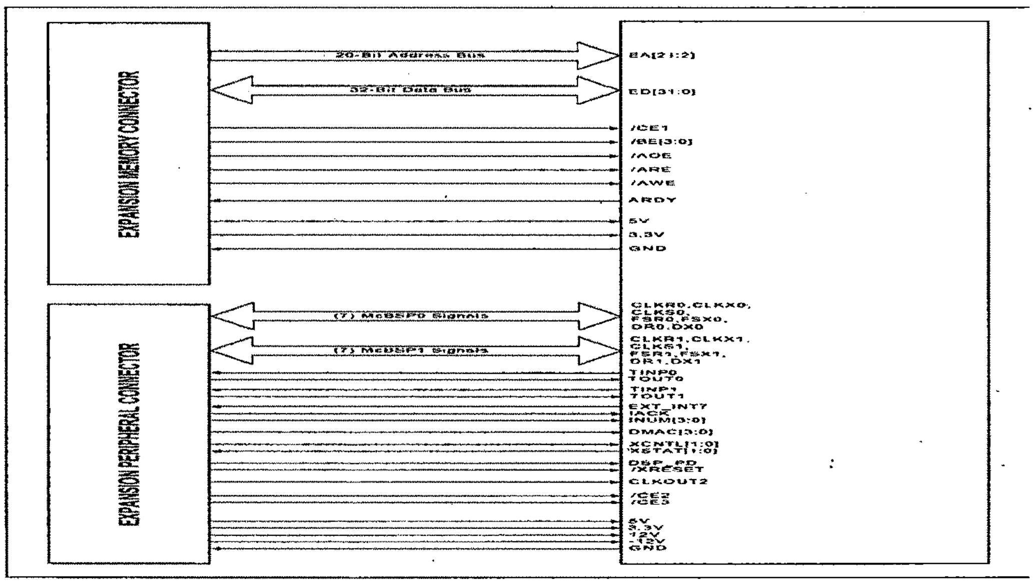


Fig 8.3 Interfacing daughter Board

8.1.2.1 Expansion Memory Interface

The expansion connectors provide access to the DSP asynchronous external memory interface (EMIF). The EMIF connector includes:

1. 20 EXTERNAL ADDRESS SIGNALS
2. 32 EXTERNAL DATA SIGNALS
3. CE1 MEMORY SPACE ENABLE
4. FOUR BYTES ENABLES
5. FOUR ASYNCHRONOUS CONTROL SIGNALS
6. POWER SIGNALS

The expansion memory interface includes:

- **20 external address signals (EA[21:2]).** All of the DSP's 20 external address signals are available on the expansion memory interface, allowing up to 4M bytes of external memory to be addressed. However, because the CE1 space must be shared with onboard EVM peripherals, only the lower 3M bytes are available to the daughterboard. If CE2 or CE3 is used for external asynchronous memory instead of SDRAM, an additional 4M bytes in each of these memory spaces can be addressed. A total of 11M bytes of asynchronous memory is directly addressable on a daughterboard. A memory page register could be used to enable a very large memory space on a daughterboard.
- **32 external data signals (ED[31:0]);** 32external data signals are available on the expansion memory interface to support full 32-bit word accesses to the daughterboard.
- **CE1 memory space enable (/CE1).** CE1 memory space enable is available on the expansion memory interface to allow asynchronous accesses to daughterboard memory and memory-mapped devices. The CE2 and CE3 memory space enables are provided on the expansion peripheral connector for additional asynchronous memory address spaces.
- **Four byte enables (/BE[3:0]).** The DSP's four byte enables are available on the expansion memory interface to support byte (8-bit), halfword (16-bit) and word (32-bit)daughterboard memory accesses.
- **Four asynchronous control signals (/AOE, /ARE, /AWE and ARDY).** The DSP's asynchronous EMIF control signals are provided to control memory accesses to the daughterboard. These signals indicate the direction of the data transfer and allow the daughterboard to add wait states to memory accesses.
- **Power signals.** The expansion memory interface also provides ground, 5-V and 3.3-V voltages to the daughterboard. The expansion memory interface supports both 3.3-V and 5-V devices since it uses low-voltage translation buffers that have 5-V tolerant inputs. The expansion memory interface is provided by a dual-row, 80-pin surface-mount connector.

8.1.2.2 Expansion Peripheral Interface

The EVM's expansion peripheral interface provides the DSP's on-chip peripheral signals to a daughterboard. This peripheral interface allows synchronous serial devices such as codecs and communication controllers, to be added to the EVM via a daughterboard. The expansion peripheral interface includes:

1. Seven signals for each of the serial ports
2. Two I/O signals for each of the DSP timers
3. Interrupt, interrupt acknowledge, and identification signals

4. Four DMA completion flags
5. Four general-purpose input/output flags
6. Power-down signal
7. Reset signal
8. Buffered CE2 and CE3 signals
9. Power signals

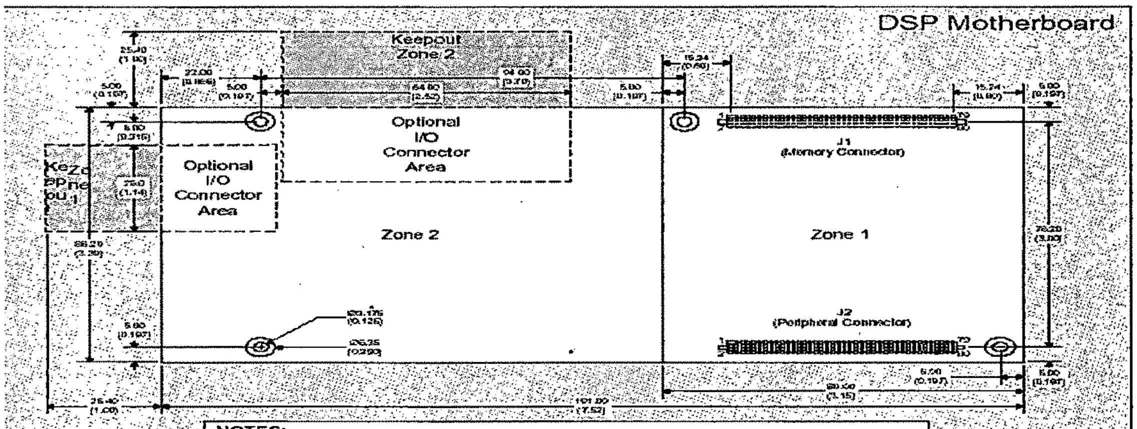
- **McBSP1** signals available on the expansion peripheral interface are buffered by a TI 'CBT3384 device to support both 5-V and 3.3-V serial devices using McBSP1 on the daughterboard.
- **McBSP0** signals are also available when the DSP software controls onboard TI 'CBT3257 multiplexers that connect them to the expansion connector rather than the onboard audio codec. This architecture provides a daughterboard with access to both of the DSP's serial ports, which is useful in many DSP applications. Due to multiplexer, both 5-V and 3.3-V serial devices can use McBSP0 on a daughterboard. Both serial ports allow full-duplex connections between the DSP and the daughterboard consisting of separate data, frame sync and clock signals for the receive and transmit paths.
- **input/output signals for DSP timers (timer 0 / timer 1).** The expansion peripheral interface includes each of the DSP timers' input and output signals. This allows timer signals to be sent to the daughterboard, or timer input or events to be counted to come from the daughterboard. Each timer has one input and one output signal.
- **Interrupt, interrupt acknowledge, and identification signals.** A DSP external interrupt (EXT_INT7) is included on the expansion peripheral interface to allow the daughterboard to interrupt the DSP to notify it of data transfers and other significant events. This interrupt is pulled down on the EVM, so the daughterboard must drive it high to interrupt the DSP (rising edge). Additionally, the DSP's interrupt acknowledge (IACK) and interrupt identification number signals (INUM[3:0]) are available to the daughterboard.
- **The DMA action complete flags (DMAC[3:0)** are available to the daughterboard on the expansion peripheral interface. These signals provide a method of feedback to external logic generating an event for each DMA channel. The DMAC signals can also be used for general-purpose output control signals controlled from the DSP's DMA channel secondary control register.
- **Four general-purpose input/output flags.** Two general-purpose control outputs and status inputs are brought to the expansion peripheral interface to allow the DSP to control and monitor various signals on a daughterboard. The XCNTL[1:0] and XSTAT[1:0] signals can be controlled by DSP software by accessing the CPLD's DSP memory-mapped CNTL and STAT registers respectively.
- **Power-down signal.** The DSP's power-down indication signal (DSP_PD) is also brought to the expansion peripheral interface so that a daughterboard can be powered-down if desired.
- **Reset signal.** The expansion peripheral interface also provides a reset signal that is active low when the board is in the reset state. This allows circuitry on the daughterboard to be set to a known state. The reset signal is asserted for a minimum of 140 ms upon power-up, via a manual reset pushbutton on the EVM or under host or DSP software control. A memory-mapped register bit in the CPLD's CNTL register allows DSP software to directly control this reset signal.

- **CLKOUT2 clock signal.** The DSP's CLKOUT2 signal (CPU clock divided by 2) is brought out to the expansion peripheral interface for synchronization needs on daughterboards.
- **Buffered CE2 and CE3 signals.** The DSP's CE2 and CE3 memory space enables are brought out to the expansion peripheral interface to support additional fast memory decoding. This can be useful on daughterboards that have multiple devices that need memory decodes. The CE2 and CE3 are dedicated to SDRAM use on the EVM board, but the EMIF control registers can be initialized for asynchronous operation, which disables the respective SDRAM banks and allows expansion asynchronous memory to be used instead. The CE2 and CE3 SDRAM enable bits in the CPLD's SDCNTL register must be disabled to allow CPLD logic to enable the external data transceivers for CE2/CE3 daughterboard accesses.
- **Power signals.** The expansion peripheral interface also provides ground, 12-V, -12-V, 5-V and 3.3-V power signals to the daughterboard.

The serial synchronous ports can run up to 80 Mbits/sec (10 Mbytes/sec) with a 160 MHz DSP, so they are about an order of magnitude slower than the parallel, memory-mapped interface. Serial interfaces do provide the advantage of less complexity, with only a handful of signals being required. The serial ports provide a direct data connection to the DSP without contending for EMIF bandwidth. Several industry-standard devices such as computer telephony switches (MVIP, SCSA, H.110), T1/E1 framers, A/D converters and D/A converters can be directly connected to a McBSP.

8.1.2.3 Cross platform Specifications

A standard has been established for daughter cards made to function with TMS320C6000™ systems. The daughter card specification is based on the interfaces provided with several existing DSP motherboard TMS320C6711 DSK.


DSP Motherboard
Keepout Zone 2
Optional I/O Connector Area
Optional I/O Connector Area
Zone 2
Zone 1
J1 (Memory Connector)
J2 (Peripheral Connector)

NOTES:
1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.
2. Drawing shows component side of DSP board.
3. Full-size daughtercard area is 191 x 86.2 mm (7.52 x 3.39 in).
4. Daughtercard interface connectors are Samtec SFM-140-L2-S-D-LC.

Fig 8.4 Daughter card Interface Layout on DSP Motherboard

The facilities present on the motherboards and the layout of a daughter card is specified such that a card may reside within a PC chassis when attached to a PCI motherboard, though the interface is not restricted to PCI platforms. Signals are brought to the daughter card through two 80-pin headers, with one header primarily for peripheral signals and the other primarily for the external memory interface. The daughter card interface on the DSP motherboard has several set requirements concerning signal drive, timing delay, and voltage tolerance. It is important to design logic, perform timing analysis, and develop software drivers considering the differences between the DSP interfaces and development boards [Evaluation Modules (EVMs) and DSP Starter Kits (DSKs)]. The daughter card interfaces on all existing DSP motherboards consist of two 80-pin headers.

Pin	**Signal**	**Description**	**Pin**	**Signal**	**Description**
1	12V	12V Voltage Supply Pin	2	-12v	12V Voltage Supply Pin
3	GND	System Ground	4	GND	System Ground
5	5V	5V Supply pin	6	5V	5V Supply pin
7	GND	System Ground	8	GND	System Ground
9	5V	5V Supply pin	10	5V	5V Supply pin
11-17	NC	No Connect	12-18	NC	No Connect
19	3.3V	3.3 Voltage supply pin.	20	3.3V	3.3 Voltage supply pin.
21	CLKX0	McBSP0 transmit Clock	22	NC	No connect
23	FSK0	McBSP0 transmit frame sync.	24	NC	No connect
25	GND	System Ground	26	GND	System Ground
27	CLKR0	McBSP0 receive Clock	28	NC	No connect
29	FSKR0	McBSP0 receive frame sync.	30	DR0	McBSP0 receive data
31	GND	System Ground	32	GND	System Ground
33	CLKX1	McBSP1 transmit Clock	34	NC	No connect
35	FSK1	McBSP1 transmit frame sync.	36	DR0	McBSP1 receive data
37	GND	System Ground	38	GND	System Ground
39	CLKR1	McBSP1 receive Clock	40	NC	No connect
41	FSKR1	McBSP1 receive frame sync.	42	DR1	McBSP1 receive data
43	GND	System Ground	44	GND	System Ground
45	TOUT0	Timer0 output	46	TIN0	Timer0 input
47	NC	NO connect	48	Ext_int5	External Interrupt 5
49	TOUT1	Timer1 Output	50	TIN1	Timer1 input
51	GND	System Ground	52	GND	System Ground
53	EXT_INT4	Ext. Interrupt 4	54	NC	No connect
55	NC	No connect	56	NC	No connect
57	NC	No connect	58	NC	No connect
59	RESET	System Reset	60	NC	No connect
61	GND	System Ground	62	GND	System Ground
63	CNTL1	Daughter Card control 1	64	CNTL0	Daughter Card control 0
65	STAT1	Daughter Card status 1	66	STAT0	Daughter Card status 0
67	EXT_INT6	Ext. Interrupt 6	68	EXT_INT7	Ext. Interrupt 7
69-73	NC	No connect	70-74	NC	No connect
75	GND	System Ground	76	GND	System Ground
77	GND	System Ground	78	ECLKOUT	EMIF Clock
79	GND	System Ground	80	GND	System Ground

Table 8.1 TMS320C6711 DSK Peripheral Connector (J2): Pinouts

Pin	Signal	Description	Pin	Signal	Descripion
1	5V	5V Voltage Supply Pin	2	5 V	5 V Voltage Supply Pin
3	EA21	EMIF Add Pin 21	4	EA20	EMIF Add Pin 20
5	EA19	EMIF Add Pin 19	6	EA18	EMIF Add Pin 18
7	EA17	EMIF Add Pin 17	8	EA16	EMIF Add Pin
9	EA15	EMIF Add Pin 15	10	EA14	EMIF Add Pin 14
11	GND	System Ground	12	GND	System Ground
13	EA13	EMIF Add Pin 13	14	EA12	EMIF Add Pin 12
15	EA11	EMIF Add Pin 11	16	EA10	EMIF Add Pin 10
17	EA9	EMIF Add Pin 0	18	EA8	EMIF Add Pin 8
19	EA7	EMIF Add Pin 7	20	EA6	EMIF Add Pin 6
21	5 V	5V Voltage Supply pin	22	5 V	5V Voltage Supply pin
23	EA5	EMIF Add Pin 5	24	EA4	EMIF Add Pin 4
25	EA3	EMIF Add Pin 3	26	EA2	EMIF Add Pin 2
27	BE3#	EMIF Byte Enable 3	28	BE2#	EMIF Byte Enable 2
29	BE1#	EMIF Byte Enable 1	30	BE0#	EMIF Byte Enable 0
31	GND	System Ground	32	GND	System Ground
33	ED31	EMIF Data pin 31	34	ED30	EMIF Data pin 30
35	ED29	EMIF Data pin 29	36	ED28	EMIF Data pin 28
37	ED27	EMIF Data pin 27	38	ED26	EMIF Data pin 26
39	ED25	EMIF Data pin 25	40	ED24	EMIF Data pin 24
41	3.3 V	3.3 V Power Supply pin	42	3.3 V	3.3 V Power Supply pin
43	ED23	EMIF Data pin 31	44	ED22	EMIF Data pin 22
45	ED21	EMIF Data pin 29	46	ED20	EMIF Data pin 20
47	ED19	EMIF Data pin 27	48	ED18	EMIF Data pin 18
49	ED17	EMIF Data pin 25	50	ED16	EMIF Data pin 16
51	GND	System Ground	52	GND	System Ground
53	ED15	EMIF Data pin 15	54	ED14	EMIF Data pin 14
55	ED13	EMIF Data pin 13	56	ED12	EMIF Data pin 12
57	ED11	EMIF Data pin 11	58	ED10	EMIF Data pin 10
59	ED9	EMIF Data pin 9	60	ED8	EMIF Data pin 8
61	GND	System Ground	62	GND	System Ground
63	ED7	EMIF Data pin 7	64	ED6	EMIF Data pin 6
65	ED5	EMIF Data pin 5	66	ED4	EMIF Data pin 4
67	ED3	EMIF Data pin 3	68	ED2	EMIF Data pin 2
69	ED1	EMIF Data pin 1	70	ED0	EMIF Data pin 0
71	GND	System Ground	72	GND	System Ground
73	ARE#	EMIF Async. RD_ En	74	AWE#	EMIF Async. WR_ En
75	AOE#	EMIF Async. OUT_ En	76	ARDY#	EMIF Async. ready
77	CE3#	Chip Enable 3	78	CE2#	Chip Enable 2
79	GND	System Ground	80	GND	System Ground

Table 8.2 TMS320C6711 DSK Memory Connector (J1): Pinouts

ne header contains the memory interface signals, while the other contains peripheral signals. The dimensions of the daughtercard layout area are designed to facilitate being incorporated on a PCI card within a PC chassis. The interface is not intended to be restricted to such platforms, however. Any DSP board can be equipped with a compliant daughtercard interface provided that the signal pinout and physical requirements are met. The interface connectors on theTMS320C6711 DSK (Fig 8.4) is shown in the Table 8.1 and Table 8.2. Table 8.3 gives summary of Daughter Card Interface Features (DSK TMS320C6711)

Memory Interface	Width	8/16/32
	Addressing	Byte
	Access	Byte
		16-Bit word
		32-Bit word
	Chip Select	2
	Memory Types	Async
	Addressable Mmeory	CE2: 4 MB
		CE3: 4 MB
Serial Ports		2 McBSP
Timers		2 TINP
		2 TOUT
Interrupts	DC to DSP	4
	DSP to Dc	0
DSP System Status		RESET
GPIO		2 Input
		2 Output
Host I/F		None

Table 8.3 Daughter Card Interface Features (DSK TMS320C6711)

- **Memory Interface**: C6711 DSK provide an asynchronous memory interface capable of communicating with 8-, 16-, or 32-bit memory. There are two chip selects. The interface is byte addressable, with 20 address lines and individual byte enables. The memory is controlled with separate output, read, and write enables. A ready input to the DSP is available to indicate a memory is not ready. When using 8-bit or 16-bit memory the physical connection of the memory depends on the endianness of the DSP. For little endian mode of operation, the memory is attached to the low numbered data lines. For big endian mode, the memory is attached to the high-numbered data lines. The addressable memory for both memory spaces offerthe full 4MB of memory to the daughtercards: DC_EA[21:2] = 0x000000 – 0xFFFFF (byte address range = 0x000000 – 0x3FFFFF).

- ***Serial Ports*** All of the interfaces documented provide a serial interface to the daughtercard.

- The C6711 DSK provide two 7-signal serial ports to the daughtercard. This includes clock, frame, and data signals for both the transmit and receive data streams, as well as a clock input to operate the serial port asynchronously to the DSP. One serial interfaces is dedicated to the daughter card and while the other is selectable between the daughter rcard and system hardware. The serial port signals may be used as general-purpose I/O

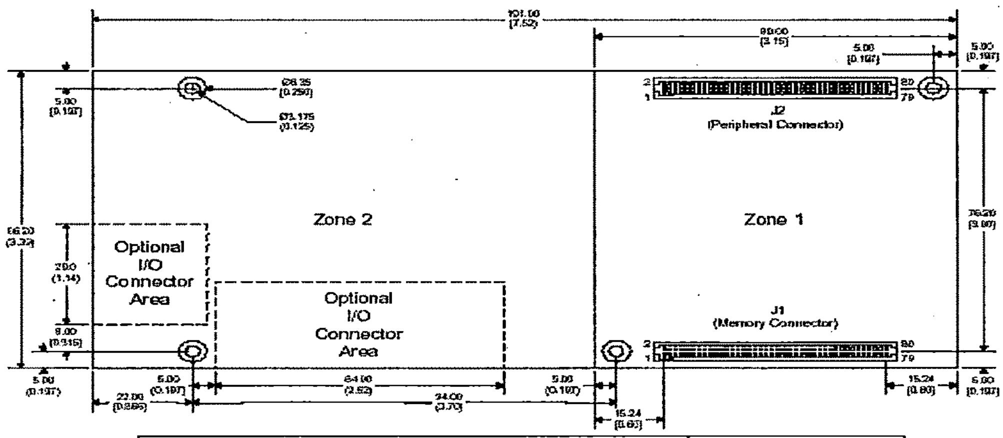
- ***Timers*** The on-chip timers are made available to the daughtercard on **C6711 DSK,** Each of the two timers available consist of an input signal and an output signal. The input can be used as either a general purpose input or as an event to be counted internally. Output signal can either be a general-purpose output, a periodic pulse, or a clock output.

- Interrupt: The C6711 DSK provides four external interrupts for the daughtercard. It does not provide the non-maskable interrupt.
- ***Status and Control Signals:*** C6711 DSKs do not have the status indicators that are present on the C6201. The DSKs provide only the reset signal to the daughtercard.
- ***General-Purpose I/O*** Several of the interfaces have general-purpose signals present to communicate between the DSP (or system) and the daughtercard. C6711 DSK have two general-purpose inputs and two outputs for communication. These are referred to as status and control signals, respectively.

8.1.2.4 Layout Daughter boards

The daughtercard interface consists of two signal connectors that present both an external memory interface as well as numerous peripheral signals. The physical dimensions of the daughtercard allow mounting the card on a PCI DSP motherboard while allowing the attached motherboard and daughtercard to fit within a single PCI slot of a PC. While the specification is not restricted to PCI applications, the physical dimensions of the daughtercard must fit within the maximum dimensions outlined below to be compliant.

A daughtercard may be designed to use some or all of the signals presented by the daughtercard interfaces on the DSP motherboards. Likewise, a daughtercard may be physically any size that fits within the maximum dimension of 191 x 86.2mm. Mounting holes are provided on both the daughtercard and the DSP board to allow stability. The daughtercard layout showing the component side is shown in Fig 8.5.



NOTES:
1. All dimensions are shown in millimeters. Inch dimensions are shown in parentheses.
2. Drawing shows component side of daughtercard.
3. Full-size daughtercard area is 191 x 86.2 mm (7.52 x 3.39 in).
4. Daughtercard connectors are Samtec TFM-140-32-S-D-LC (or TFM-140-31-D-LC).

Fig 8.5 Daughter card Interface Layout

8.1.3 ADS 8364 EVM

The ADS8364 includes six, 16-bit, 250KHz ADCs (Analog to Digital Converters) with 6 fully differential input channels grouped into two pairs for high-speed simultaneous signal acquisition. Inputs to the sample-and-hold amplifiers are fully differential and are maintained differential to the input of the ADC to provide common-mode rejection of 80dB at 50 KHz that is important in high-noise environments. The ADS8364 offers a flexible high-speed parallel interface with a direct address mode, a cycle, and a FIFO mode. The output data for each channel is available as a 16-bit word.

Features of ADS 8364 are: 6 Input Channels with facility for differential inputs and buffered reference, 16-bit resolution 16 Bits with μS total throughput/ Channel Provides a direct connection to C5000 and C6000 DSK platforms through the 80-pin as shown in Fig 8.6

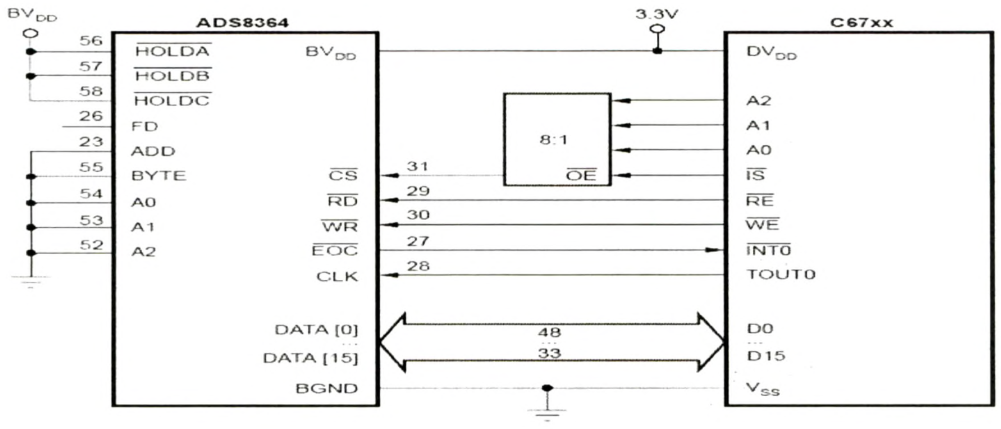


Fig 8.6 INTERFACING ADS8364 WITH 671X

The ADS8364 is a high-speed, low power, dual 16-bit A/D converter that operates from independent 5-V AVDD and DVDD supplies. The digital output is delivered through a built-in buffer circuit that can be powered from DVDD or separate 2.7-V to 5.25-V (BVDD) sources which gives flexibility for mixed voltage environments design. Conversion time for the ADS8364 is 3.2 μs when a 5-MHz external clock is used. The corresponding acquisition time is 0.8 μs. The maximum output rate is 250 KSPS.

The six fully differential sample and hold circuits are divided into three pairs (A, B, and C). Each pair of channels has a hold signal (HOLDA, HOLDB, and HOLDC) which, when strobe together, allows simultaneous sampling on all six analog inputs. The part accepts an analog input voltage in the range of –VREF to +VREF, centered on the internal 2.5-V reference. The part also accepts bipolar input ranges when a level shift circuit is used in the analog front-end circuitry.

The HOLDx signals, ADD pin control, and reset are all derived from the GPIO function of McBSP port 1. These signals are located on J12. The address and data lines are available on J11. The EVM is factory configured to use address 0xA000 0020 as its base. An alternate address base can be selected by removing the shunt jumper on W17. This assigns A17 or 0xA002 0000 as the base address of the EVM. In this configuration, channels A1 through C1 are located at the base address + 0x4000.

The ADS8364 features a byte mode in which data can be read from the ADC in two consecutive 8-bit reads. W16 controls the byte feature, which is disabled by default. To enable byte mode, remove the jumper from W16. The BNC connector (J9) and W15 allow for the selection of an external conversion clock source. Factory default settings provide the clock source via the TOUT1 signal of the DSP by placing a shunt jumper on pins 1 and 2 of W15. By moving the shunt to pins 2 and 3, the EVM user can apply an external clock source of not more than 5 MHz to J9.

The power supply requirements for the ADS8364EVM board can be split into three categories—analog front end, ADC power, and digital interface. While filters are provided for all power supply inputs, optimal performance of the EVM requires a clean, well-regulated power source. The power and ground planes on the inner layers of the EVM are split into digital and analog sections, with the ground planes tied together at a single point near the filter circuits.

8.1.4 TLV 2553 EVM

The TLV2553 is a 12-bit, switched-capacitor ADC using SA The ADC has three control inputs [chip select (CS)\, the input-output clock, and the address/control input (DATAIN)], designed for communication with the serial port of a host processor or peripheral through a serial 3-state output. The device has an on-chip 14-channel multiplexer that can select any one of 11 inputs or any one of three internal self-test voltages using configuration register 1. The sample-and-hold function is automatic. At the end of conversion, when programmed as EOC, the pin 19 output goes high to indicate that conversion is complete. The converter incorporated in the device features differential, high-impedance reference inputs that facilitate ratio metric conversion, scaling, and isolation of analog circuitry from logic and supply noise. A switched-capacitor design allows low-error conversion over the full operating temperature range.

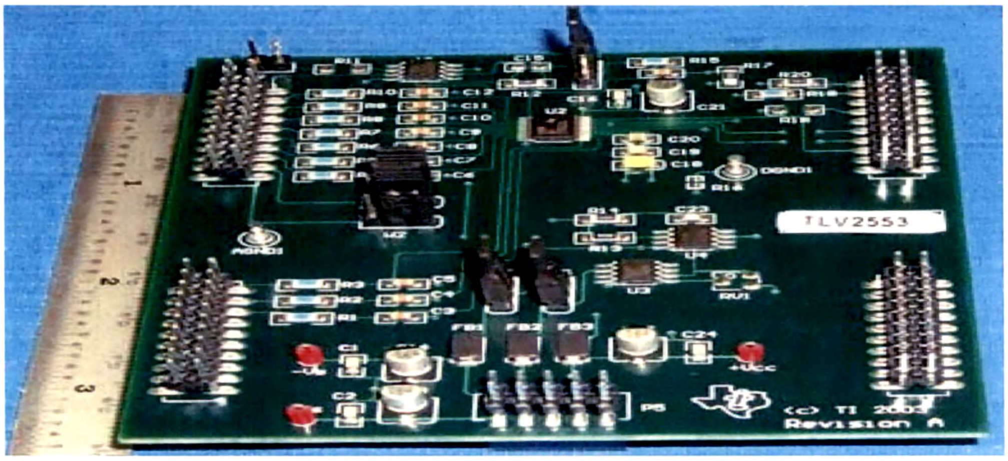


Fig 8.7 TLV 2553 EVM BOARD

The TLV2553EVM (fig 8.7) is 11-Channel, 12-bit analog-to-digital converter board based on the TLV2553 ADC. The ADC uses a synchronous serial interface which can be simply interfaced to many micro controllers using the SPI protocol. The EVM also incorporates a stable voltage reference buffered by an operational amplifier, to ensure a low-noise voltage reference for the ADC. A block diagram for the analog interface of the EVM is shown in Fig 8.8 .The channels are arranged to comply with the EVM standard developed for data converters. This standard defines eight channels of analog I/O on each EVM module.

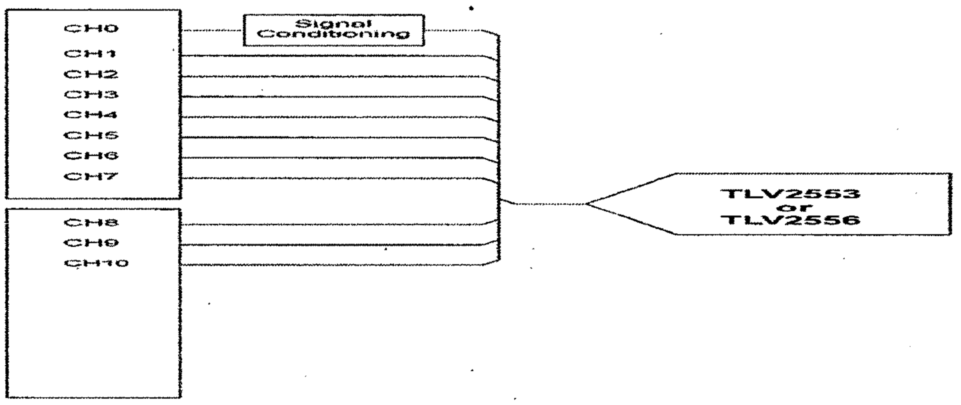


Fig 8.8 ANALOG INTERFACE BLOCK DIAGRAM

The amplifier present on the EVM operates from a dual power supply and is configured with a gain of 1. If signal conditioning is not required, it can easily be bypassed by a shorting bar at W6.

8.1.5 TLV 5614 EVM

The TLV5614 is a quadruple 12-bit voltage output digital-to-analog converter (DAC) with a flexible 4-wire serial interface. The 4-wire serial interface allows glue less interface to TMS320, SPI, QSPI, and Micro wire serial ports. The TLV5614 is programmed with a 16-bit serial word comprised of a DAC address, individual DAC control bits, and a 12-bit DAC value. The device has provision for two supplies: one digital supply for the serial interface (via pins DV_{DD} and DGND), and one for the DACs, reference buffers, and output buffers (via pins AV_{DD} and AGND). Each supply is independent of the other, and can be any value between 2.7 V and 5.5 V. The dual supplies allow a typical application where the DAC will be controlled via a microprocessor operating on a 3 V supply (also used on pins DV_{DD} and DGND), with the DACs operating on a 5 V supply. Of course, the digital and analog supplies can be tied together.

The resistor string output voltage is buffered by a x2 gain rail-to-rail output buffer. The buffer features a Class AB output stage to improve stability and reduce settling time. A rail-to-rail output stage and a power-down mode makes it ideal for single voltage, battery based applications. The settling time of the DAC is programmable to allow the designer to optimize speed versus power dissipation. The control bits within the 16-bit serial input string choose the settling time.

This EVM (Fig 8.9) features four independent digital-to-analog converters on one convenient evaluation platform. The EVM is available in a 12-bit version. The EVM contains a precision 5-V reference for the DACs, as well as connection terminals for an independent external reference. The analog outputs from each DAC are available on separate pin headers. Each DAC is address selectable through hardware or user supplied software. The EVM has a built-in test mode that allows the user to input a digital word to the DAC and verify the conversion result.

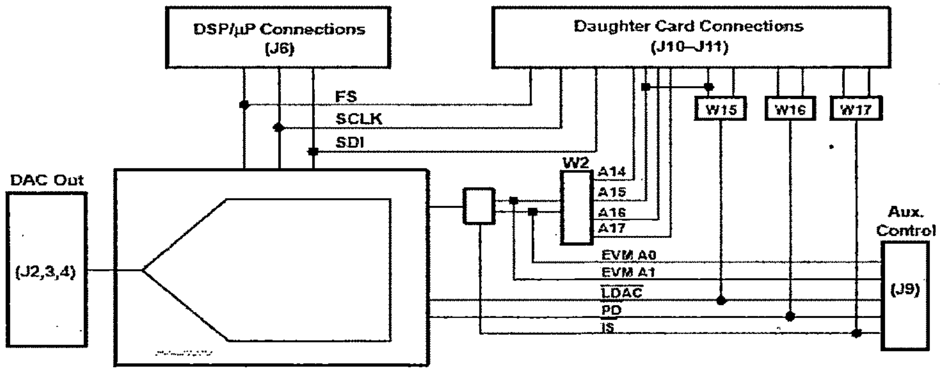


Fig 8.9 EVM BLOCK DIAGRAM

Closing W13 activates an onboard 20 MHz clock. This puts the EVM in self-test mode. The onboard oscillator provides the SCLK signal to the data converters and test circuitry. Hardware jumpers W10, W12, and W14 allow the EVM user to select the DAC for evaluation. W10 and W12 control U3 and U4 respectively. Opening these jumpers allows the frame sync signal to be routed to the data converter. W14 provides a chip select signal to U5 (pulls CS low). One, two, or all three DACs can be operated at the same time.

The onboard clock is fed through a 4-bit binary counter whose terminal count output is delayed one clock cycle and used as the DAC frame sync. In test mode, frame sync is common to all DACs. The test mode frame sync signal is inverted and used as the parallel-load enable signal to U10. This inverted signal also clears the binary counter (U9). The two 8-channel data converters (U3 and U4) on this EVM support daisy-chaining of the serial input data. Jumpers W6 and W9 support the daisy chain devices. Jumper W11 allows the EVM user to configure the serial data source.

The EVM uses four banks of 4-bit switches to create a 16-bit serial word to the data converters. This is accomplished through the use of parallel-load serial shift registers U8 and U10. The internal configuration registers of the DACs can be set, and serial data can be simulated.

When operated outside of the test mode (W13 open), the DACs expect to receive their control signals and serial data input from a host processor. The EVM works with TI's DSK series of digital signal processor evaluation boards that support the common connector interface. Connectors J10 and J11, located on the underside of the EVM, provide the DSP interface. Connectors J6, J7, and J8 allow the EVM user to define a custom processor interface. The shorting bars on J7 and J8 can be removed, allowing the EVM user to interface to older DSKs, micro controllers, or pattern generators.

EVM ADDRESS BUS: Address decoder U13 determines access to each data converter. A two-bit address bus on the EVM can be accessed via J9, or daughterboard connector J10. For TMS320C6000 systems using the common connector, jumper block W2 allows the user to select any two of the high order address bits A14, A15, A16 or A17.

The DAC EVM is designed to operate from 3.3 V to 5.5 V. The EVM requires 170 mA at 3.3 V and 150 mA at 5.5 V. Power to the EVM can be applied via J1 from an external supply, or can be directed through the common connector interface via J10. Jumper W1 allows the EVM user to select either the 3.3-V or 5-V bus from the DSP.

8.1.6 5 – 6 K INTERFACE EVM

The interface board Fig (8,10) consists of two signal conditioning sites, two serial EVM sites, and a parallel EVM site. All EVMs compatible with the 5-6K Interface Board have a standard analog interface and standard power connector. Three position screw terminals J1 and J2 and two position screw terminals J6, J8, J9, and J11 provide access to a common power bus routed to all sites. The Interface Board maintains a compatible interface with the TMS320 Series of Digital Signal Processors (DSP). The EVM is a Circuit board which allows the designer to mate the board with TI DSK.

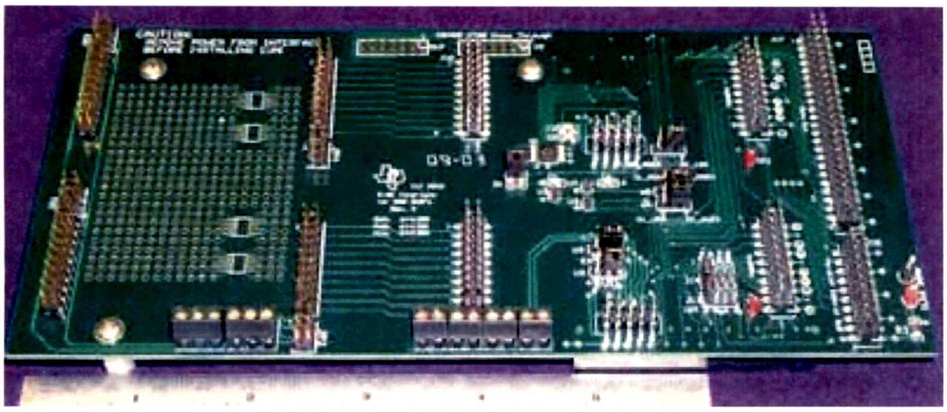


Fig 8.10 5K – 6 K INTERFACE CARD

8.2 Software Development Tools

The software tools described in this section include:

- Real Time Workshop®
- Code Composer STudio
- MATLAB Link for Code Composer Studio
- Data Converter Plug-Ins
- SIMULINK Blocks: Embedded Targets

8.2.1 Real-Time Workshop®

Real-Time Workshop® is an extension of capabilities found in Simulink® and MATLAB® to enable rapid prototyping of real-time software applications on a variety of systems. RTW generates optimized, portable, and customizable ANSI C code from Simulink models to create stand-alone implementations of models that operate in real-time and non-real-time in a variety of target environments. Generated code can run on PC hardware, DSPs, microcontrollers on bare-board environments, and with commercial or proprietary real-time operating systems (RTOS). Real-Time Workshop lets you speed up simulations, build in intellectual property protection, and operate across a wide variety of real-time rapid prototyping targets. Fig 8.11 illustrates the role of Real-Time Workshop in the software design process

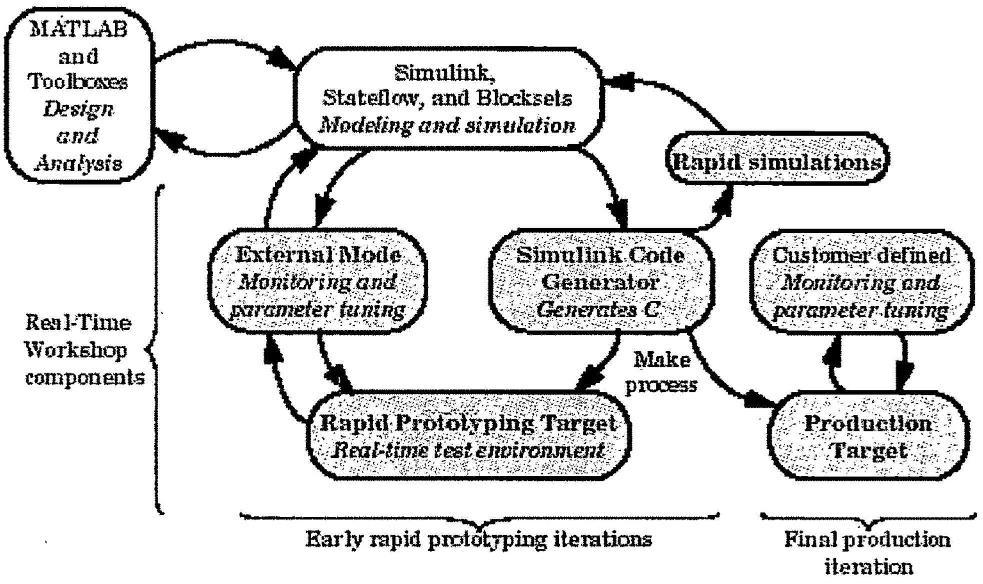

MATLAB and Toolboxes Design and Analysis
Simulink, Stateflow, and Blocksets Modeling and simulation
Rapid simulations
External Mode Monitoring and parameter tuning
Simulink Code Generator Generates C
Customer defined Monitoring and parameter tuning
Real-Time Workshop components
Make process
Rapid Prototyping Target Real-time test environment
Production Target
Early rapid prototyping iterations
Final production iteration

Fig 8.11 SOFTWARE DEVELOPMENT Using MATLAB/SIMULINK

The optional Real-Time Workshop Embedded Coder works with Real-Time Workshop to generate efficient, embeddable source code. Fig 8.12 depicts the process of generating source code from SIMULINK models.

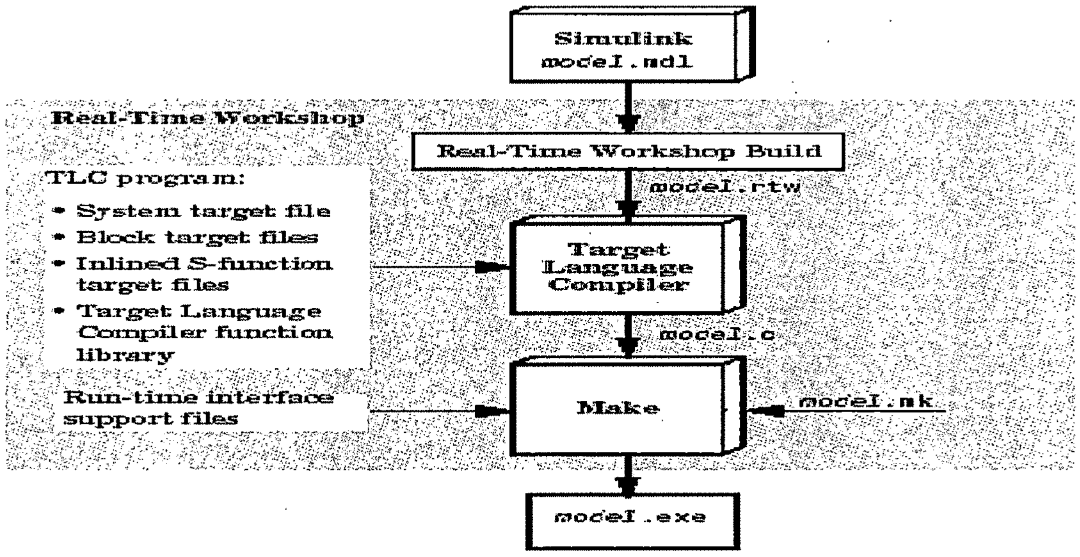
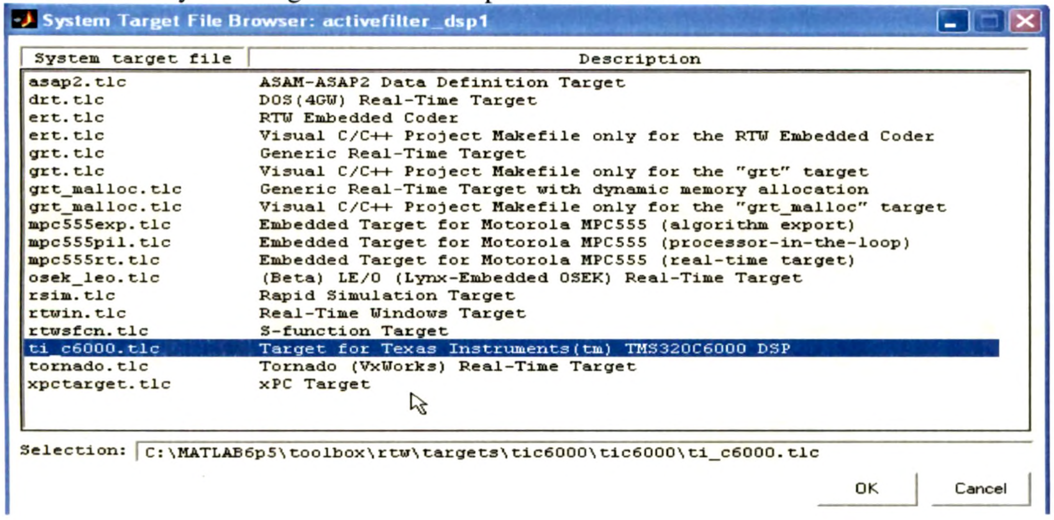


Fig 8.12 RTW Code Generation

A *target* is an environment—hardware or operating system—on which your generated code will run. The process of specifying this environment is called *targeting*. The process of generating target-specific code is controlled by a *system target file*, a *template make file*, and a *make* *command*. To select a desired target, specify these items individually, or choose from a wide variety of ready-to-run configurations.

The *host* is the system used to run MATLAB, SIMULINK, and RTW. Using the build tools on the host, code and an executable that runs on target system can be created These include ready-to-run configurations and third-party targets. Fig 8.13 gives a complete list of bundled targets, with their associated system target files and template make files.



System target file	Description
asap2.tlc	ASAM-ASAP2 Data Definition Target
drt.tlc	DOS(4GW) Real-Time Target
ert.tlc	RTW Embedded Coder
ert.tlc	Visual C/C++ Project Makefile only for the RTW Embedded Coder
grt.tlc	Generic Real-Time Target
grt.tlc	Visual C/C++ Project Makefile only for the "grt" target
grt_malloc.tlc	Generic Real-Time Target with dynamic memory allocation
grt_malloc.tlc	Visual C/C++ Project Makefile only for the "grt_malloc" target
mpc555exp.tlc	Embedded Target for Motorola MPC555 (algorithm export)
mpc555pil.tlc	Embedded Target for Motorola MPC555 (processor-in-the-loop)
mpc555rt.tlc	Embedded Target for Motorola MPC555 (real-time target)
osek_leo.tlc	(Beta) LE/O (Lynx-Embedded OSEK) Real-Time Target
rsim.tlc	Rapid Simulation Target
rtwin.tlc	Real-Time Windows Target
rtwsfcn.tlc	S-function Target
ti_c6000.tlc	Target for Texas Instruments(tm) TMS320C6000 DSP
tornado.tlc	Tornado (VxWorks) Real-Time Target
xpctarget.tlc	xPC Target

Selection: C:\MATLAB6p5\toolbox\rtw\targets\tic6000\tic6000\ti_c6000.tlc

Fig 8.13 TARGET FILE BROWSER

Real-Time Workshop provides a generic real-time development target. The GRT target provides an environment for simulating fixed-step models in single or multitasking mode. A program generated with the GRT target runs your model, in simulated time, as a stand-alone program on your workstation. The GRT target allows to perform code validation by logging system outputs, states, and simulation time to a data file. The data file can then be loaded into the MATLAB workspace for analysis or comparison with the output of the original model. The GRT target also provides a starting point for targeting custom hardware. The GRT harness program grt_main.c can be modified to execute code generated from the model at interrupt level under control of a real-time clock.

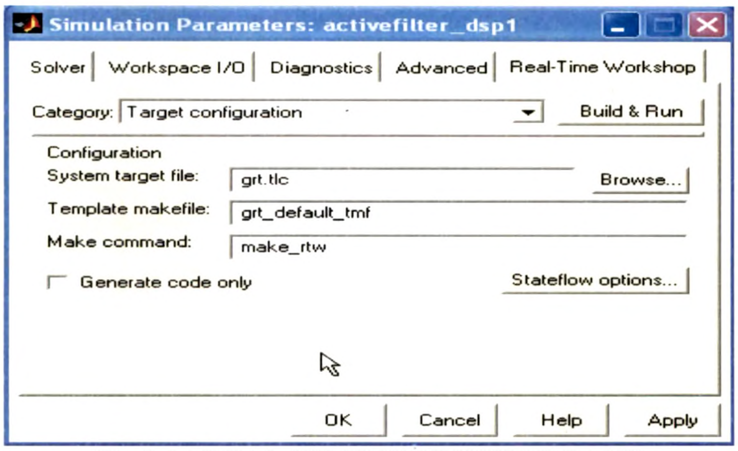
Real-Time Workshop generates source code for models and blocks through the Target Language Compiler, which reads script files (or *TLC files)* that specify the format and content of output source files.

Two types of TLC files are used:

- A *system target file*, which describes how to generate code for a chosen target, is the entry point for the TLC program that creates the executable.
- *Block target files* define how the code looks for each of the SIMULINK blocks in the model.

Real-Time Workshop uses template make files to build an executable from the generated code. The Real-Time Workshop build process creates a make file from the template make file. Each line from the template make file is copied into the make file; tokens encountered during this process are expanded into the make file. The name of the make file created by the build process is *model*.mk (where *model* is the name of the Simulink model).

The *model*.mk file is passed to a make utility, which compiles and links an executable from a set of files. A template make file has an extension of .tmf and a name corresponding to your target and compiler. System target files and template make files using the Real-Time Workshop pane of the **Simulation Parameters** dialog box (Fig 8.14) can be specified either by typing their filenames or choosing them with the Target File Browser.



Category: Target configuration

System target file: grt.tlc

Template makefile: grt_default_tmf

Make command: make_rtw

☐ Generate code only

Fig 8.14 REAL TIME WORKSHOP PANE

A high-level M-file command controls the Real-Time Workshop build process. The default command, used with most targets, is make_rtw. When you initiate a build, Real-Time Workshop invokes make_rtw.

The make_rtw command, in turn, invokes the Target Language Compiler and utilities such as make. The build process consists of the following stages:

- make_rtw compiles the block diagram and generates a model description file, *model*.rtw.
- make_rtw invokes the Target Language Compiler to generate target-specific code, processing *model*.rtw as specified by the selected system target file.
- make_rtw creates a make file, *model*.mk, from the selected template make file.
- make is invoked which compiles and links a program from the generated code, as instructed in the generated make file.

8.2.2 S-function

An S-function is to create custom Simulink blocks. An advantage of using S-functions is that we can build a general purpose block that can use many times in a model, varying parameters with each instance of the block. S-function is use to make generalize Simulink model of the system. S-functions can be written in MATLAB, C, C++. S-functions use a special calling syntax that enables to interact with Simulink equation solvers. The interaction is very similar to the interaction that takes place between the solvers and built-in Simulink blocks. The form of an S-function is very general and can accommodate continuous, discrete, and hybrid systems. S-functions allow to add new blocks to Simulink models. S-functions can also be used with the Real-Time Workshop to customize the code generated by the Real Time Workshop for S-functions by writing a Target Language Compiler (TLC) file.

8.2.2.1 M-FILE S-FUNCTIONS

An M-file S-function consists of a MATLAB function of the following form:

[sys,x0,str,ts]=f (t, x, u, flag, p1, p2) : where f is the S-function's name, t is the current time, x is the state vector of the corresponding S-function block, u is the block's inputs, flag indicates a task to be performed, and p1, p2, ... are the block's parameters.

8.2.2.2 FLAG ARGUMENT

Table 8.4 gives the value of flag and its association with the handle in the simsize structure.

Flag	S-Function Routine	Description
Flag =0	mdlInitializeSizes	Defines basic S-Function block characteristics, including sample times, initial conditions of continuous and discrete states, and the sizes array.
Flag =1	mdlDerivatives	Calculates the derivatives of the continuous State variables.
Flag =2	mdlUpdate	Updates discrete states, sample times, and major time step requirements
Flag =3	mdlOutputs	Calculates the outputs of the S-function.
Flag =4	mdlGetTimeOfNextVarHit	Calculates the time of the next hit in absolute time. (This routine is used only when you specify a variable discrete-time sample time in mdlInitializeSizes).
Flag =9	mdlTerminate	Performs any necessary end-of-simulation tasks.

Table 8.4 Flag Characteristics

8.2.2.3 S-Function BLOCK CHARACTERISTICS

For Simulink to recognize an M-file S-function, must provide it with specific information about the S-function. This information includes the number of inputs, outputs, states, and other block characteristics as depicted in Table 8.5

Field Name	Description
sizes.NumContStates	Number of continuous states
sizes.NumDiscStates	Number of discrete states
sizes.NumOutputs	Number of outputs
sizes.NumInputs	Number of inputs
sizes.DirFeedthrough	Flag for direct feed through
sizes.NumSampleTimes	Number of sample times

Table 8.5 Fields in SIZE Structure

To pass information to SIMULINK the following steps may be used:

1. call the simsizes function at the beginning of mdlInitializeSizes ➔ **sizes = simsizes;**
2. call simsizes again using ➔ **sys = simsizes(sizes);**

8.2.3 Code Composer Studio

Fig 8.15 depicts the procedure to generate code. The CCStudio IDE provides a graphical interface for using the code generation tools. A CCStudio project keeps track of all information needed to build a target program or library. A project records:

- Filenames of source code and object libraries
- Compiler, assembler, and linker options
- Include file dependencies

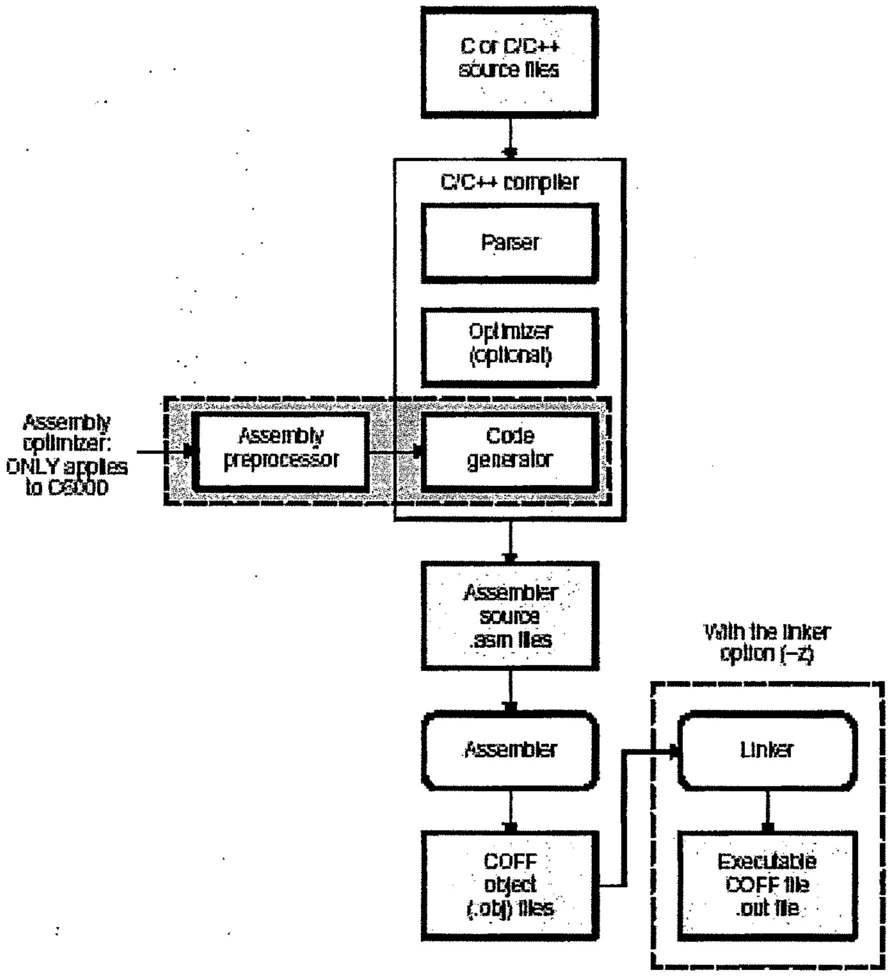


Fig 8.15 CODE DEVELOPMENT FLOW

The CCStudio IDE supports all phases of the development cycle shown in Fig 8.16

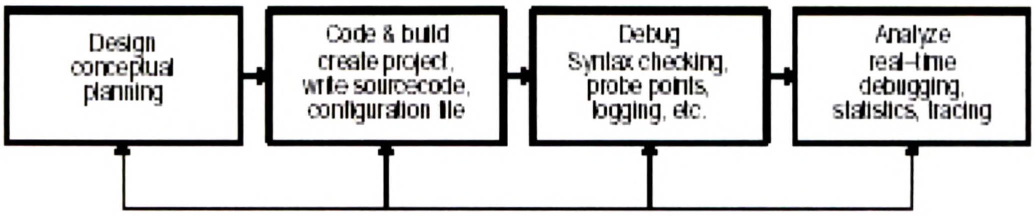
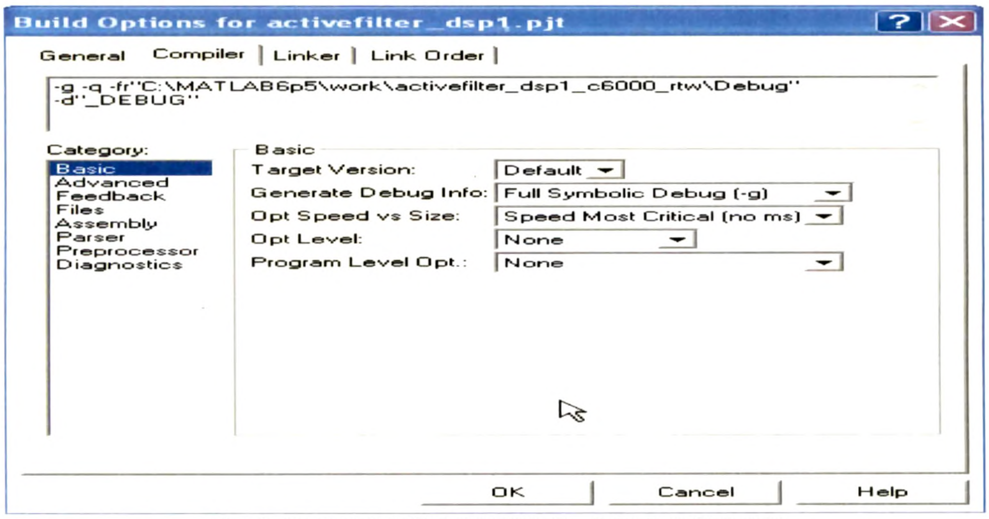


Fig 8.16 SIMPLIFIED CCSSTUDIO DEVELOPMENT FLOW

The compiler, assembler, and linker options can be specified within CCS's Build Options dialog box (Fig 8.17). Nearly all command line options are represented within this dialog box. Options that are not represented can be specified by typing the option directly into the editable text box that appears at the top of the dialog box.



-g -q -fr"C:\MATLAB6p5\work\activefilter_dsp1_c6000_rtw\Debug"
-d"_DEBUG"

Category:
Basic
Advanced
Feedback
Files
Assembly
Parser
Preprocessor
Diagnostics

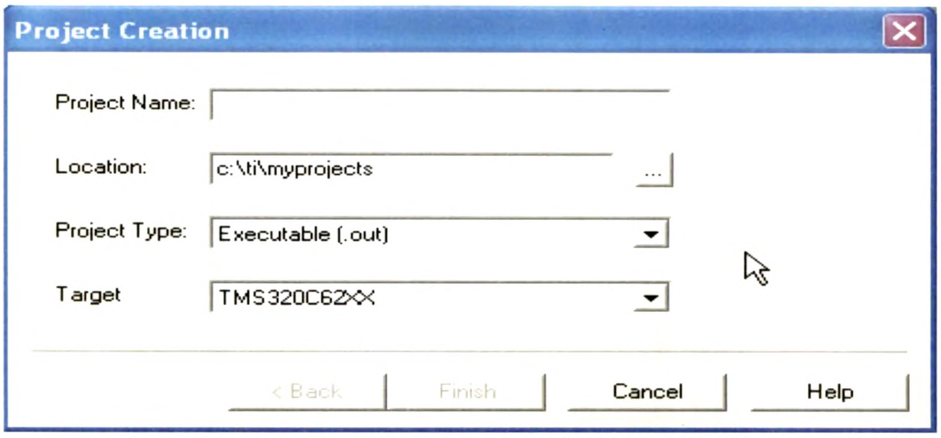
Basic

Target Version:	Default
Generate Debug Info:	Full Symbolic Debug (-g)
Opt Speed vs Size:	Speed Most Critical (no ms)
Opt Level:	None
Program Level Opt.:	None

Fig 8.17 BUILDS OPTIONS: DIALOG BOX

The following procedure allows user to create new projects, either individually or several at once. Each project's filename must be unique. The information for a project is stored in a single project file (*.pjt).

- From the Project menu, choose New. The Project Creation wizard window (Fig 8.18) displays.



Project Name:

Location: c:\ti\myprojects

Project Type: Executable (.out)

Target TMS320C62XX

Fig 8.18 PROJECT CREATION WINDOW

- In the Project Name field, type the name you want for your project.
- In the Location field, specify a directory to store the project file. You can type the full path in the Location field or click the Browse button and use the Choose Directory dialog box.
- In the Project Type field, select a Project Type from the drop-down list. Choose either Executable (.out) or Library (lib). Executable indicates that the project generates an executable file. Library indicates that you are building an object library.
- In the Target field, select the Target Family that identifies your CPU. This information is necessary when tools are installed for multiple targets.
- Click Finish.

The CCStudio IDE creates a project file (Fig 8.19) called *projectname*.pjt, which stores project settings and references the various files used by the project. The new project automatically becomes the active project. The first project configuration (in alphabetical order) is set as active. The new project inherits TI-supplied default compiler and linker options for debug and release configurations.

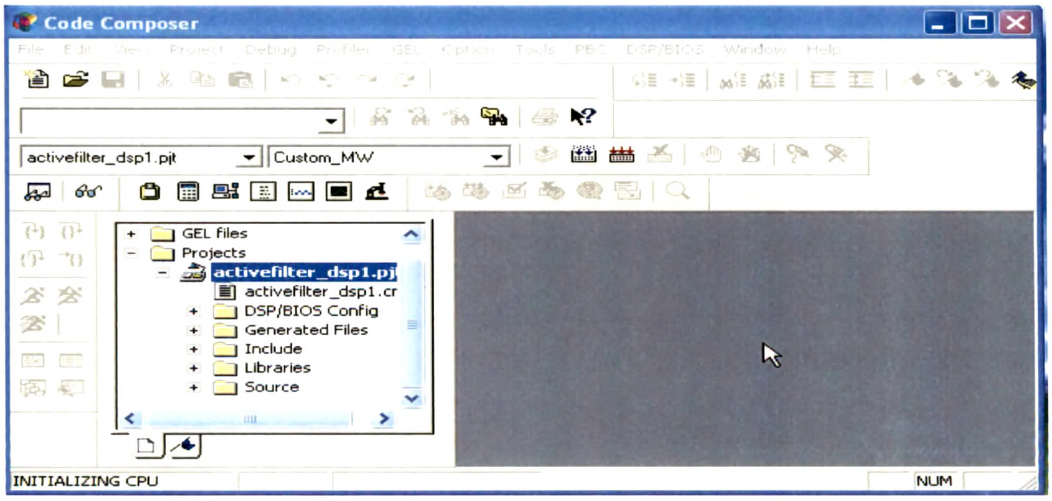
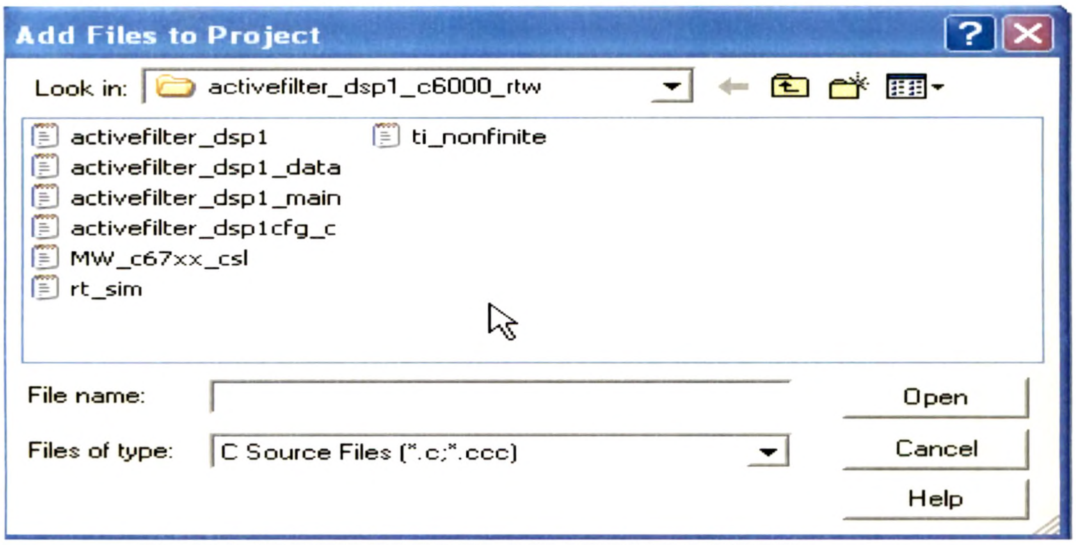


Fig 8.19 CCSSTUDIO IDE BASIC WINDOW

After creating a new project file, add the filenames of the source code, object libraries, and linker command file to the project list. The procedure to add files to the project:

- Select Project → Add Files to Project, or right-click on the project's filename in the Project View window and select Add Files. The Add Files to Project dialog box displays.(Fig 8.20)



Add Files to Project

Look in: activefilter_dsp1_c6000_rtw

activefilter_dsp1
activefilter_dsp1_data
activefilter_dsp1_main
activefilter_dsp1cfg_c
MW_c67xx_csl
rt_sim
ti_nonfinite

File name:

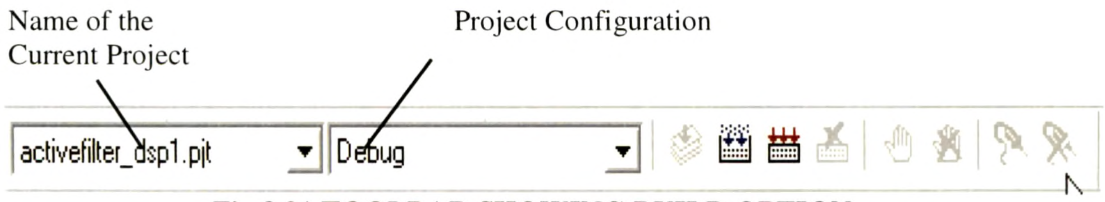
Files of type: C Source Files (*.c;*.ccc)

Fig 8.20 ADD FILES TO PROJECT DIALOG BOX

- In the Add Files to Project dialog box, specify a file to add. If the file does not exist in the current directory, browse to the correct location. Use the Files of type drop-down list to set the type of files that appear in the File name field.

- Click Open to add the specified file to your project. The Project View is automatically updated when a file is added to the current project. The project manager organizes files into folders for source files; include files, libraries, and DSP/BIOS configuration files. Source files that are generated by DSP/BIOS are placed in the Generated files folder. To build and run a program, follow these steps:

- Choose Project → Rebuild All or click the (Rebuild All) toolbar button. The CCStudio IDE recompiles, reassembles, and relinks all the files in your project. Messages about this process are shown in a frame at the bottom of the window.
- By default, the .out file is built into a debug directory located under your current project folder. To change this location, select a different one from the CCStudio toolbar.(Fig 8.21)



activefilter_dsp1.pjt

Debug

Fig 8.21 TOOLBAR SHOWING BUILD OPTION

- Choose File → Load Program. Select the program you just rebuilt, and click Open. The CCStudio IDE loads the program onto the target DSP and opens a Dis-Assembly window that shows the disassembled instructions that make up the program. (Notice that the CCStudio IDE also automatically opens a tabbed area at the bottom of the window to show the output that the program sends to stdout.)
- Choose View → Mixed Source/ASM. This allows you to simultaneously view your c source and the resulting assembly code.
- Click on an assembly instruction in the mixed-mode window. (Click on the actual instruction, not the address of the instruction or the fields passed to the instruction.) Press the F1 key. The CCStudio IDE searches for help on that instruction. This is a good way to get help on an unfamiliar assembly instruction.
- Choose Debug → Go Main to begin execution from the main function. The execution halts at main and is identified by
- Choose Debug → Run or click the (Run) toolbar button to run the program.
- Choose Debug → Halt to quit running the program.

8.2.4 MATLAB Link for Code Composer Studio

The link provides Development Tools to use MATLAB functions to communicate with Code Composer Studio™ and with information stored in memory and registers on a target. With the links user can transfer information to and from CCS and with the embedded objects. get The information about data and functions stored in the signal processor memory and registers, as well as information about functions in the project can be obtained. The links and the embedded objects are objects which can be used like all other MATLAB objects. The object properties and their methods can be set, changed or manipulated. The link allows creating two kinds of objects:

- Links that connect MATLAB to Code Composer Studio
- Embedded objects created by user to provide access to data and functions in the ccs and target. The link objects allow to use the embedded objects to access the target.

C6000 target in Real-Time Workshop®, the MATLAB Link for Code Composer Studio supports: TMS320C6701 EVM, TMS320C6711 DSK and C6xxx simulator.Links for RTDX and CCS work with any board that CCS supports. The link provides three components that work with and use CCS IDE and TI Real-Time Data Exchange (RTDX™):

• **Link for Code Composer Studio IDE:** To use objects to create links between CCS IDE and MATLAB ®. From the command window, you can run applications in CCS IDE, send to and receive data from target memory, and check the processor status, as well as other functions such asstarting and stopping applications running on your digital signal processors.

•**Link for Real-Time Data Exchange Interface**: It provides a communications pathway between MATLAB and digital signal processors installed on the PC. The objects in the MATLAB Link for Code Composer Studio are used to open channels to processors on boards in the computer and send and Retrieve data about the processors and executing applications, as well as send data to the processes for use and get data from the applications.

•**Embedded Objects**: It provides object methods and properties that let you access and manipulate information stored in memory and registers on digital signal processors, or in your Code Composer Studio project. From MATLAB you gather information from you project, work with the information in MATLAB, doing things like converting data types, creating function declarations, or changing values, and return the information to your project—all from the MATLAB command line.

To verify that CCS is installed on the system, enter **>> ccsboardinfo** … at the MATLAB command line. With CCS installed and configured, MATLAB returns information about the boards that CCS recognizes on your machine, in a form similar to the following listing:

```
Board  Board                              Proc   Processor   Processor
Num    Name                               Num    Name        Type
--- --------------------------------- --- ------------------------------
1     C6xxx Simulator (Texas Instrum ...   0     6701        TMS320C6701
0     C6x11 DSK (Texas Instruments)        0     CPU         TMS320C6x1x
```

With the target code loaded, MATLAB Link can be used for CCS Studio functions to examine and modify data values in the processor. In the CCS IDE Project view window, there should be a file named ccstut.c. The MATLAB Link for Code Composer Studio provides three functions to control target execution—run, halt, and restart.

MATLAB Link for Code Composer Studio implements just this sort of access and manipulation capability by using MATLAB objects (called embedded objects in this guide) that access and represent variables and data embedded in your project. Various functions that compose the MATLAB Link for Code Composer Studio, such as createobj, convert, and write, helps to create the embedded objects to be used to work with the data in DSP memory and registers, and allows the to manipulate the data in MATLAB as well as in user code

The MATLAB Link for Code Composer Studio and the links for CCS IDE and RTDX speed and enhance ability to develop and deploy digital signal processing systems on TI DSPs. MathWorks tools, CCS IDE and RTDX work together to test and analyze processing algorithms in the MATLAB workspace. In contrast to CCS IDE, using links for RTDX allows user to interact with the process in real time while it's running on the target. Across the link, it is possible to:

- Send and retrieve data from memory on the processor
- Change the operating characteristics of the program
- Make changes to algorithms as needed without stopping the program or setting breakpoints in the code Enabling real-time interaction to the process or algorithm in action, the results as they develop, and the way the process runs.

A data buffer is created from MATLAB is sent to DSP processor (DSK/EVM). The tools required are:
1. Embedded Target for TI C6000 DSP
2. MATLAB link for CCS.

Support files required are:

1. c671Xdsk.cmd ==Linker Command file
2. Vecor file
3. rtdx.lib == Library support file
4. A header file to enable the interrupt
5. C program creates a channel:

- input == PC-MATLAB to DSK
- output channel.=== DSK to PC-MATLAB

Input channel will read data from PC and output channel will send the result. (I/O is designated from Target)
// RTDX_matlab_sim.c ➔ MATLAB-DSK interface using RTDX between PC & DSK
// RTDX_MATLAB_sim.m ➔ MATLAB-DSK interface using RTDX. Calls CCS

8.2.5 SIMULINK Blocks: Embedded Targets

User can use Real-Time Workshop to generate, target, and execute Simulink models on the TI C6711 DSK. The software is the ideal resource for rapidly prototyping and developing embedded systems applications for the TI C6711 Digital Signal Processor. The Embedded Target software focuses on developing real-time DSP applications for the C6711 DSK. The library of simulink blocks for the C6711target is depicted in Fig 8.22

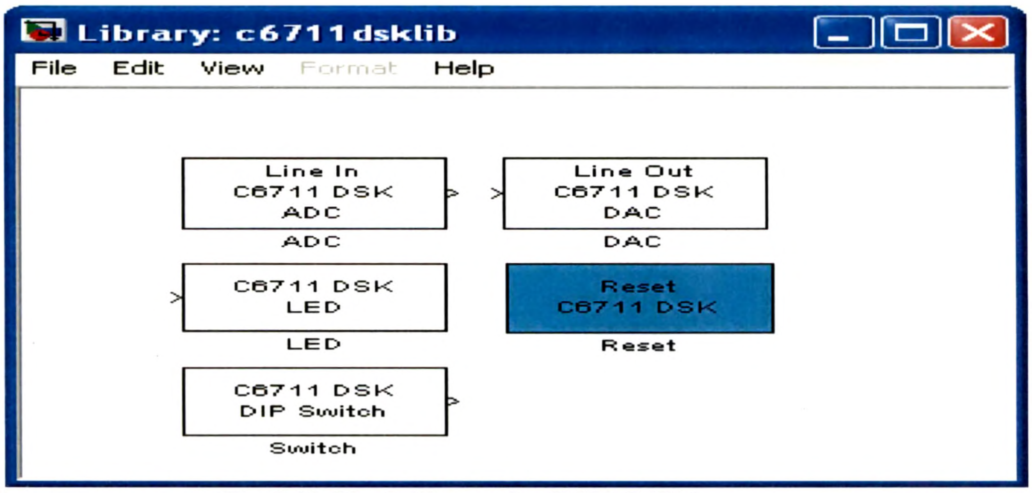


Fig 8.22 Block library for DSK C6711

The blocks provided in the library are corresponding to the I/O component such as LDE, Swithces, ADC and DAC, on the kit TMS 320C6711 DSK

- Input block : (C6711 DSK ADC)
- Output block : (C6711 DSK DAC)
- Light emitting diode block (C6711 DSK LED)
- DIP switch block (C6711 DIP Switch)
- Reset block (Reset C6711 DSK)

8.2.6 Data Converter Plug-In: DCP

Data converters provide the interface between the analog world and the digital world of the digital signal processor. Most converter ICs offer only simple conversion and has simple interfaces. This has changed recently, with more functionality moving into the converter, Converters have special features suited to particular applications, such as high-precision differential inputs. Some have enhanced digital interfaces while others execute common pre/post-processing tasks to decrease the CPU load. Texas Instruments uses the plug-in architecture to extend the development environment by gradually implementing DSP-optimized ADCs and DACs. The goal is to simplify integration of data converters from device initialization up to the sample processing into its own algorithm.The Data Converter Plug-In (DCP) is a software plug-in with the Code Composer Studio™ (CCS) for the TMS320C2800™; TMS320C5000™, and TMS320C6000™ digital signal processor (DSP) families. It generates C source code drivers based on user inputs for Texas Instruments (TI) data converters (ADC, DAC, and CODEC) connected to a TI DSP.

The Data Converter Support Software (DCP) is a useful tool. It generates driver source code in response to inputs from the user. All interface and configuration settings are made through an easy-to-use graphical user interface (GUI). The drivers have been developed and tested on the data-converter evaluation modules (EVM) that are also available from Texas Instruments. TI offers a variety of DSP architectures optimized for a wide range of applications. In addition to DSPs for motor and process controls, there are low-cost DSPs with floating-point architecture for a variety of industrial systems. To allow for high portability, the processors are often programmed in C, using the highly efficient, optimizing C-compilers.

The C6000 DSPs offer processor performance up to several thousand MIPS (million instructions per second). Those devices are mainly used in applications requiring large processing power and high data throughput, such as compression algorithms in the audio and video segment. Typical applications are central office DSL modems as well as GSM base stations, where many data channels need to be processed simultaneously. To support both increasing system complexity and shorter design cycles, new development systems had to improve in code generation as well as in efficient debugging capabilities. All development tool modules have been integrated into one development system, the Code Composer Studio ™ (CCS). User functions to configure the DSP on-chip peripherals are combined into libraries, enabling the user to execute easily the many configuration options available. Subroutines to configure and read or write ADCs and DACs are developed for all major DSP families.

The user may generate configuration data using a graphical user interface provided by a CCS plug-in. Plug-in allows the designer to choose from the many options to configure a data converter via a dialog window. Once the parameters have been selected, the tool generates the necessary functions for the C-program that are required to configure the data converter. Functions to initiate the data transfer via the ADC-DSP interface and to read ADC conversion results are also generated. The interface can be tested easily and immediately for optimal performance.

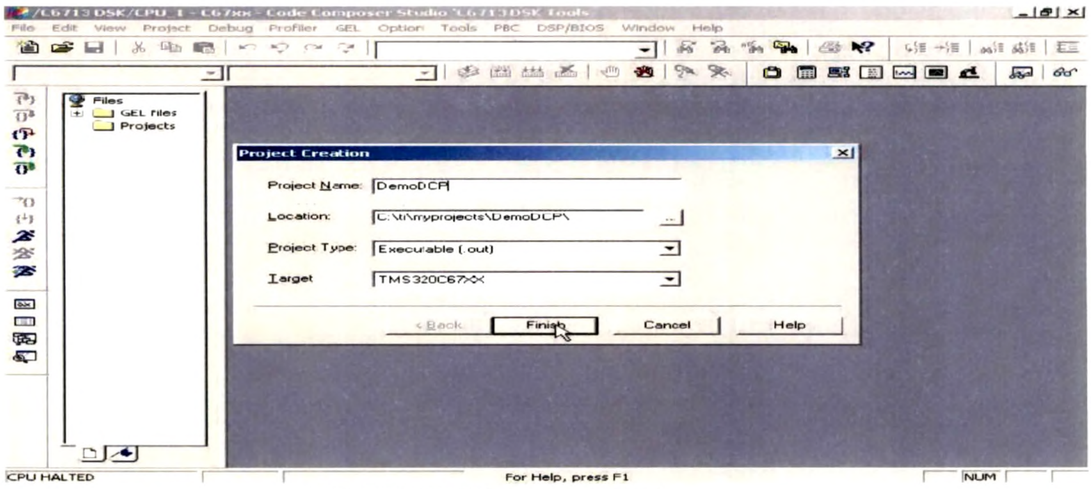
When calling the "Data Converter Support" menu option within CCS, the designer can choose from existing ADC, DAC, and AIC devices via the "Add" context menu in the dialog window, the desired converter is added to the system. Selecting "Properties" with the converter selected makes the configuration dialog window appear. This window consists of the following three segments: "Interface Settings," "ADC Settings," and "Current Control Selection." "Interface Settings," for example, includes the definition of the ADC address within the DSP memory space, the polarity of the edge of the interrupt signal, and the format of the transferred data. Within "ADC Settings," the number of analog-input channels and their operation modes is defined. The ADC conversion clock and configures the FIFO by setting the trigger level. After the source code implementation, the overall system test can begin. User can return to the "Data Converter Support" dialog window and selects new parameters. The generation of driver files requires following steps:

- Setup and Open Code Composer Studio (CCS)
- Create a New Project in CCS
- Open DCP
- Add a Converter to the DCP
- DSP Configuration
- Converter Settings
- Code Generation

8.2.6.1 CREATE A NEW PROJECT IN CCS

The DCP creates a set of files and adds them to a CCS project. Therefore, it is important to create a new project or to open an existing project before running the DCP.

- Select *CCS menu – Project – New* to create a new project, or
- Select *CCS menu* - Project – *Open* to open an existing project.


/C6713 DSK/CPU_1 - C67xx - Code Composer Studio 'C6713DSK Tools
File Edit View Project Debug Profiler GEL Option Tools PBC DSP/BIOS Window Help
Files
GEL files
Projects
Project Creation
< Back
Finish
Cancel
Help
CPU HALTED
For Help, press F1
NUM

Project Name: DemoDCP
Location: C:\ti\myprojects\DemoDCP\
Project Type: Executable (.out)
Target TMS320C67XX

Fig 8.23 CCS Project Creation

As already described in the previous chapters use the following steps to create a project:

- In the field labeled *Project Name*, type the name of the project, e.g., *DemoDCP*. (Fig 8.13)
- Verify the location where your project will be created. It is usually *c:\ti\myprojects* where *c:\ti* is the CCS installation directory.
- Set *Project Type* to *Executable* and *Target* to the DSP family (TMS320C67XX).
- Click the *Finish* button to create the project.
- CCS automatically creates a subdirectory and places the project file in it. The newly created project appears in the project pane (left side of the CCS window).

8.2.6.2 OPEN DCP

With the project open start the DCP. Select *CCS menu* - Tools - Data *Converter Support* to launch the DCP. The DCP opens with the *System Tab* selected as shown in fig 8.24.

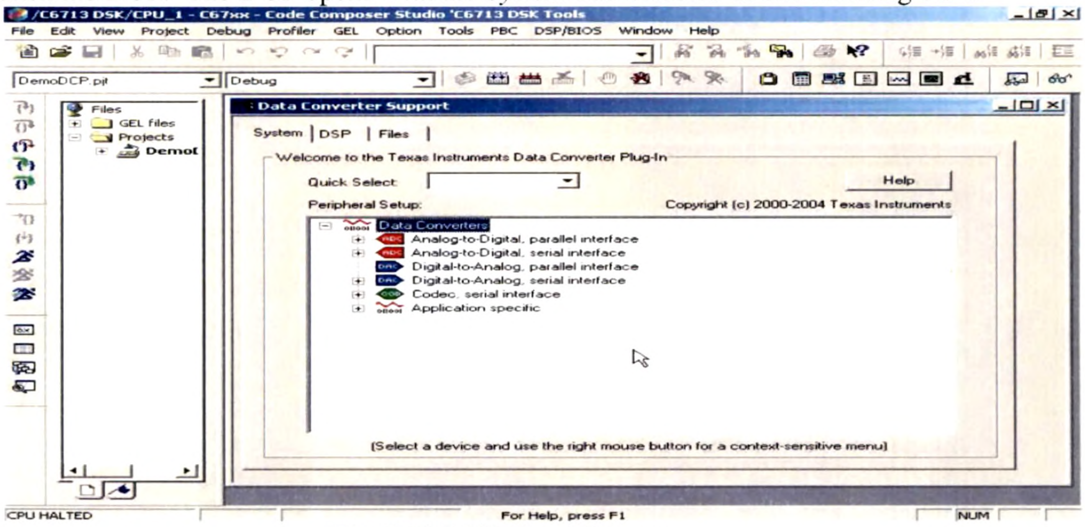

/C6713 DSK/CPU_1 - C67xx - Code Composer Studio 'C6713 DSK Tools
File Edit View Project Debug Profiler GEL Option Tools PBC DSP/BIOS Window Help
DemoDCP.pjt
Debug
Files
GEL files
Projects
Demot
Data Converter Support
System | DSP | Files
Welcome to the Texas Instruments Data Converter Plug-In
Quick Select:
Help
Peripheral Setup:
Copyright (c) 2000-2004 Texas Instruments
Data Converters
Analog-to-Digital, parallel interface
Analog-to-Digital, serial interface
Digital-to-Analog, parallel interface
Digital-to-Analog, serial interface
Codec, serial interface
Application specific
[Select a device and use the right mouse button for a context-sensitive menu]
CPU HALTED
For Help, press F1
NUM

Fig 8.24 DCP Start Screen

8.2.6.3 ADD A CONVERTER

The next step is to add the desired converter to the DCP. This can be accomplished in two ways:

- navigate the converter tree to find the desired converter based on the interface and width
- use the *Quick Select* feature, DCP searches for all converters beginning with the letters and numbers typed. Select the converter from the generated drop-down list as shown in Fig 8.25.

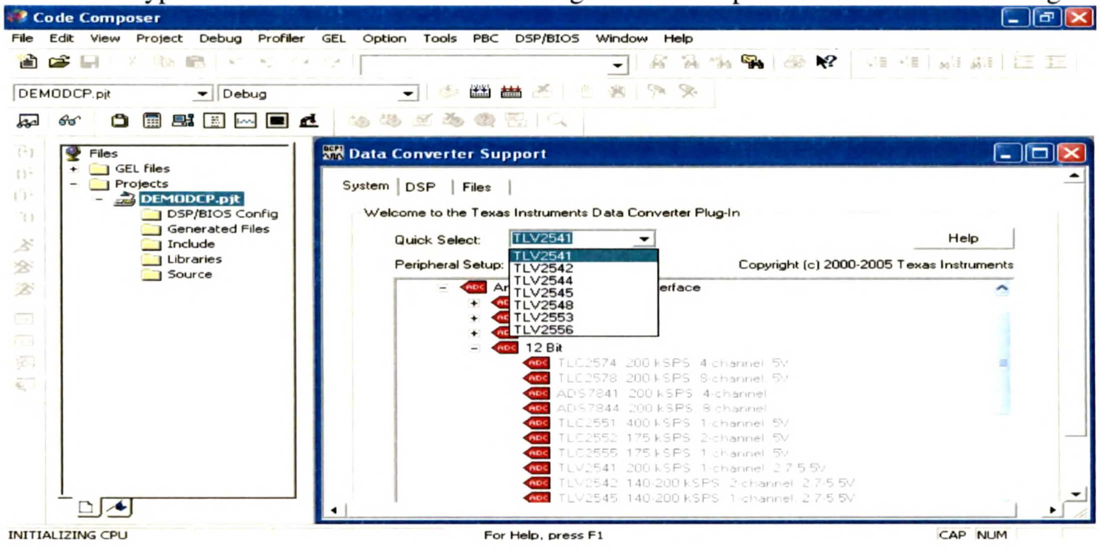
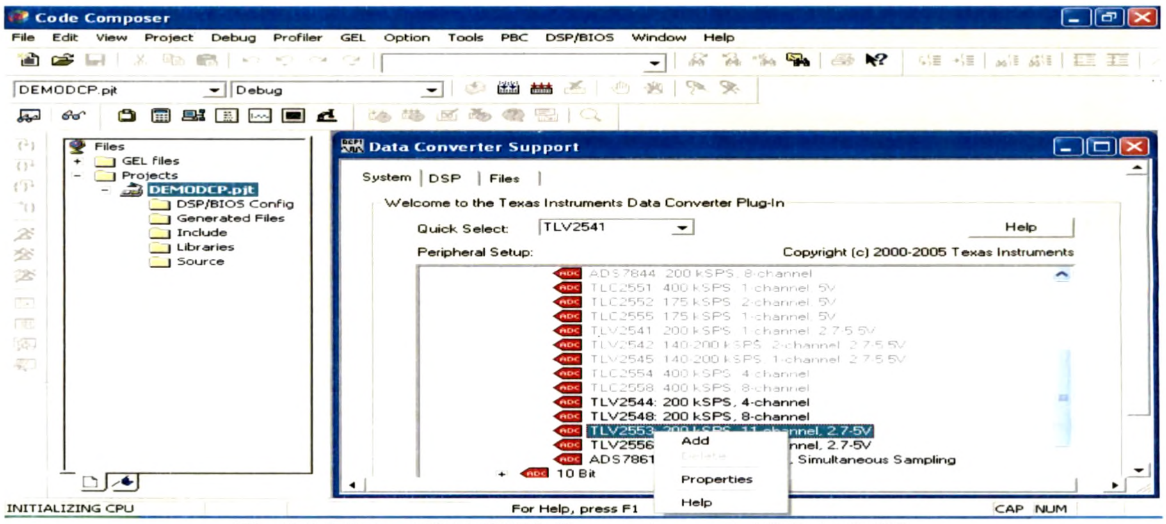

Code Composer
File Edit View Project Debug Profiler GEL Option Tools PBC DSP/BIOS Window Help
DEMODCP.pjt
Debug
Files
GEL files
Projects
DEMODCP.pjt
DSP/BIOS Config
Generated Files
Include
Libraries
Source
Data Converter Support
System | DSP | Files
Welcome to the Texas Instruments Data Converter Plug-In
Quick Select: TLV2541
Help
Peripheral Setup:
Copyright (c) 2000-2005 Texas Instruments
TLV2541
TLV2542
TLV2544
TLV2545
TLV2548
TLV2553
TLV2556
Ar erface
12 Bit
TLC2574 200 kSPS 4-channel 5V
TLC2578 200 kSPS 8-channel 5V
ADS7841 200 kSPS 4-channel
ADS7844 200 kSPS 8 channel
TLC2551 400 kSPS 1-channel 5V
TLC2552 175 kSPS 2-channel 5V
TLC2555 175 kSPS 1 channel 5V
TLV2541 200 kSPS 1-channel 2.7-5.5V
TLV2542 140-200 kSPS 2-channel 2.7-5.5V
TLV2545 140-200 kSPS 1-channel 2.7-5.5V
INITIALIZING CPU
For Help, press F1
CAP NUM

Fig 8.25 Data Converter Selection

When a converter is selected, the DCP expands the tree, highlighting the device and placing the cursor on it. Some converters are gray colored, which means that for this converter on this specific DSP platform no complete driver is available. Create the register settings for the converter, which can later be used in the driver later on.

- Header file (*dc_conf.h*) describing the converter's setup
- Abstraction layer files (*tidc_api.c, tidc_api.h*) are generated, which is described in subsequent sections.

To add the converter to the system, right-click on the highlighted converter. A pop-up menu appears (Fig 8.26). Click on the first entry labeled add, this includes the converter in the system and adds a configuration tab for it to the DCP window. Notice that a new tab called *<Converter name>*_1 (here: *TLV2553_1*) has been added to the DCP.
The pop-up menu (Fig 8.16) has two more active menu items. Clicking on the *Properties* item reveals the driver seed and driver output file names used for the selected converter. The *Help* item from the pop-up menu opens the help file of the DCP.



Welcome to the Texas Instruments Data Converter Plug-In

Quick Select: TLV2541

Peripheral Setup: Copyright (c) 2000-2005 Texas Instruments

```
ADS7844  200 kSPS, 8-channel
TLC2551  400 kSPS, 1-channel, 5V
TLC2552  175 kSPS, 2-channel, 5V
TLC2555  175 kSPS, 1-channel, 5V
TLV2541  200 kSPS, 1-channel, 2.7-5.5V
TLV2542  140-200 kSPS, 2-channel, 2.7-5.5V
TLV2545  140-200 kSPS, 1-channel, 2.7-5.5V
TLC2554  400 kSPS, 4-channel
TLC2558  400 kSPS, 8-channel
TLV2544: 200 kSPS, 4-channel
TLV2548: 200 kSPS, 8-channel
TLV2553: 200 kSPS, 11-channel, 2.7-5V
TLV2556                 nnel, 2.7-5V
ADS7861                 , Simultaneous Sampling
+ 10 Bit
```

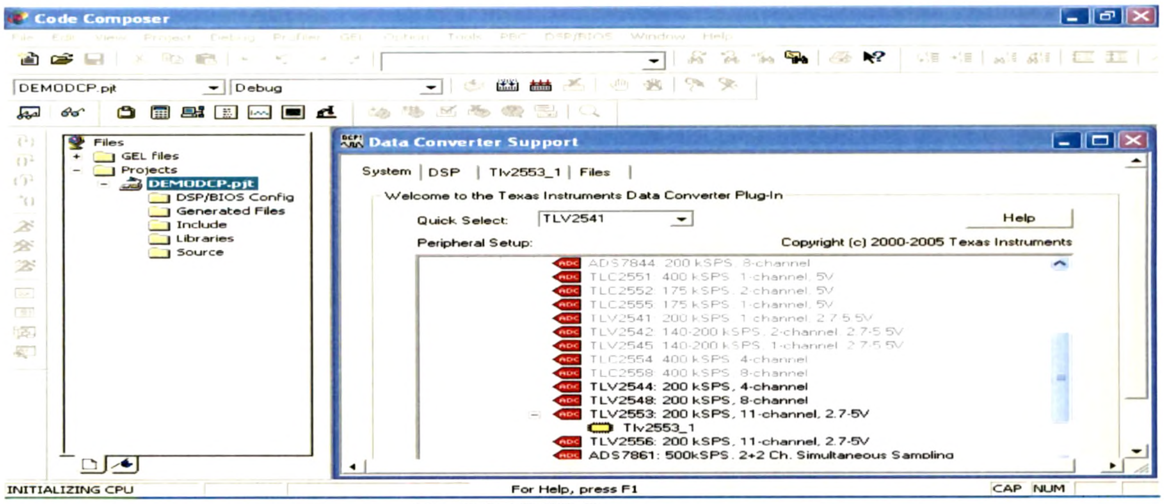
Fig 8.26 Select TLV2553 Converter and ADD to DCP

The DCP supports multiple converters in the system. If the system has more than one converter connected to the DSP, repeat the steps for all converters

8.2.6.4 DSP CONFIGURATION

The DSP-specific settings must be chosen before the converter settings can be entered. Click on the *DSP* tab to display the DSP-specific controls. The DSP page opens (Fig 8.17). Through the configuration setup, the DCP automatically determines the DSP family to be used, and lists all supported DSPs in the *DSP Type* control.

- Choose the DSP you are using in your system from the drop-down list. The *DSP Clock* control should be set to the DSP frequency used in the design. This value is used for frequency calculation for the peripherals only. It does not control the DSP frequency.
- Type the DSP frequency you are using in your system into the *DSP Clock* edit field. The *Dispatcher in DSP/BIOS Used* check box under *Misc Settings* informs the DCP if you are using the dispatcher in the DSP/BIOS. If you are not using DSP/BIOS, this checkbox must be left unchecked.



Code Composer

DEMODCP.pjt | Debug

Files
- GEL files
- Projects
  - DEMODCP.pjt
    - DSP/BIOS Config
    - Generated Files
    - Include
    - Libraries
    - Source

Data Converter Support

System | DSP | Tlv2553_1 | Files

Welcome to the Texas Instruments Data Converter Plug-In

Quick Select: TLV2541 | Help

Peripheral Setup: Copyright (c) 2000-2005 Texas Instruments

- ADS7844: 200 kSPS, 8-channel
- TLC2551: 400 kSPS, 1-channel, 5V
- TLC2552: 175 kSPS, 2-channel, 5V
- TLC2555: 175 kSPS, 1-channel, 5V
- TLV2541: 200 kSPS, 1-channel, 2.7-5.5V
- TLV2542: 140-200 kSPS, 2-channel, 2.7-5.5V
- TLV2545: 140-200 kSPS, 1-channel, 2.7-5.5V
- TLC2554: 400 kSPS, 4-channel
- TLC2558: 400 kSPS, 8-channel
- TLV2544: 200 kSPS, 4-channel
- TLV2548: 200 kSPS, 8-channel
- TLV2553: 200 kSPS, 11-channel, 2.7-5V
  - Tlv2553_1
- TLV2556: 200 kSPS, 11-channel, 2.7-5V
- ADS7861: 500kSPS, 2+2 Ch. Simultaneous Sampling

INITIALIZING CPU | For Help, press F1 | CAP NUM

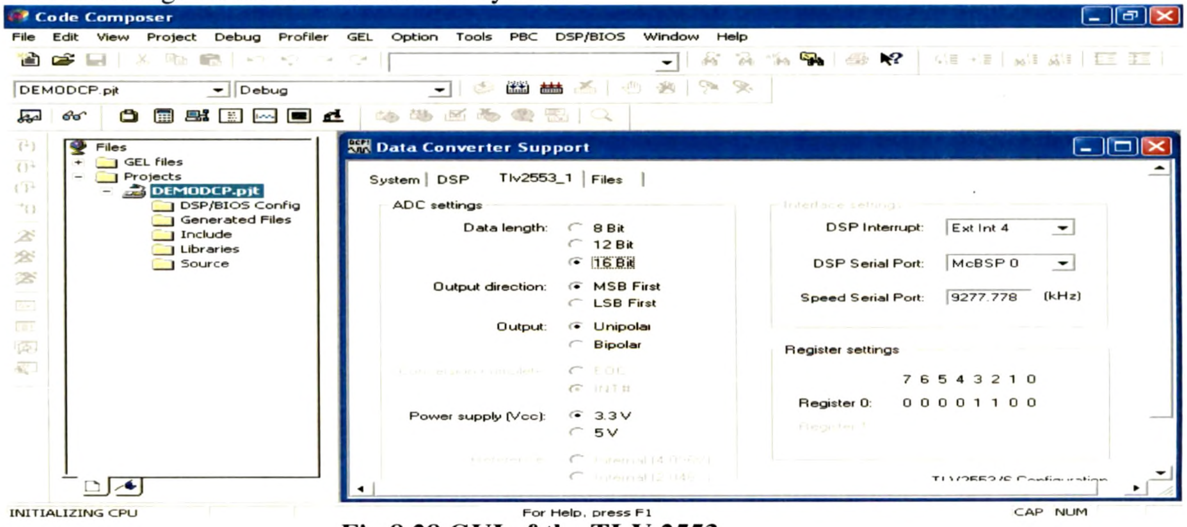
Fig 8.27 DSP Page

8.2.6.5 CONVERTER SETTINGS

Click on the tab for the data converter you added in previous section. Converters differ in capabilities and interfaces. As a result, each converter has its own unique setup tab. All pages are separated in:

- **Interface Settings**: The *Interface Settings* section is where the user enters information about the physical connection between DSP and the data converter.
- **Converter Settings (if applicable):** The *Converter Settings* section determines the behavior of the converter after power up. These settings are completely configurable—the converter works with any combination of the controls.

GUI of TLV2553 is shown in Fig 8.28. One important feature of the tool is that it is not possible to choose invalid combinations—the DCP validates all user inputs against the device constraints and displays only valid combinations for the particular converter. Choose the desired settings for all converters in the system.



Code Composer

File Edit View Project Debug Profiler GEL Option Tools PBC DSP/BIOS Window Help

DEMODCP.pjt | Debug

Files
- GEL files
- Projects
  - DEMODCP.pjt
    - DSP/BIOS Config
    - Generated Files
    - Include
    - Libraries
    - Source

Data Converter Support

System | DSP | Tlv2553_1 | Files

ADC settings

Data length: 8 Bit [✓ 16 Bit] 12 Bit

Output direction: [✓ MSB First] LSB First

Output: [✓ Unipolar] Bipolar

Power supply (Vcc): [✓ 3.3 V] 5 V

Interface settings

DSP Interrupt: Ext Int 4

DSP Serial Port: McBSP 0

Speed Serial Port: 9277.778 (kHz)

Register settings

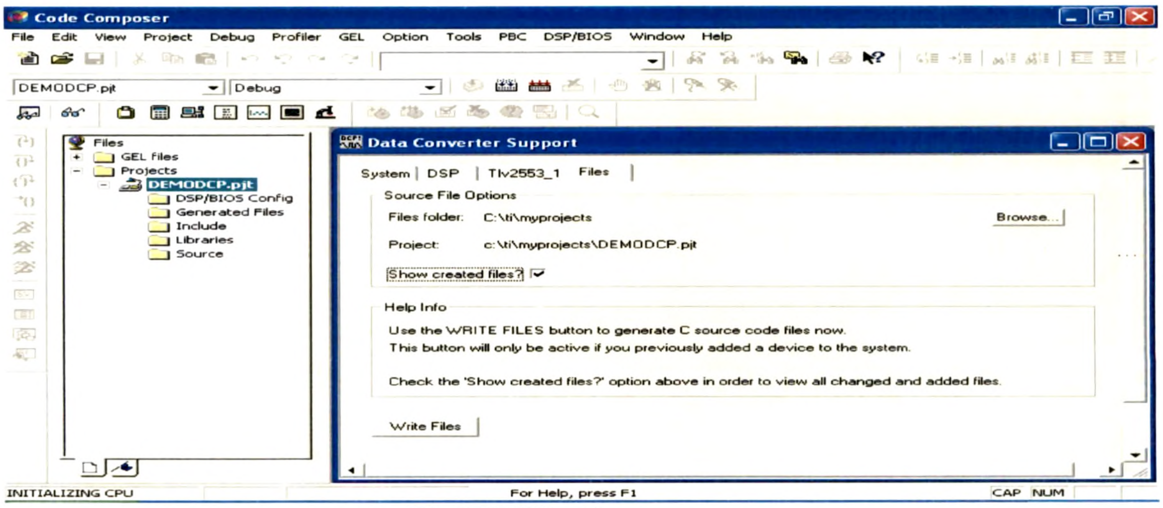
	7	6	5	4	3	2	1	0
Register 0:	0	0	0	0	1	1	0	0

INITIALIZING CPU | For Help, press F1 | CAP NUM

Fig 8.28 GUI of the TLV 2553

8.2.6.6 CODE GENERATION

Once the converter is set up, the driver code can be generated. Click on the *Files* tab (Fig 8.29). The last property page opens. The *Show Files* checkbox specifies whether the generated source files remain open (checked) or are closed (unchecked) once they are generated. Verify that directory where the driver files will be generated is the project directory. In our case the directory is named *c:\ti\myprojects\DemoDCP*.



Source File Options

Files folder: C:\ti\myprojects

Project: c:\ti\myprojects\DEMODCP.pjt

Show created files? ✓

Help Info

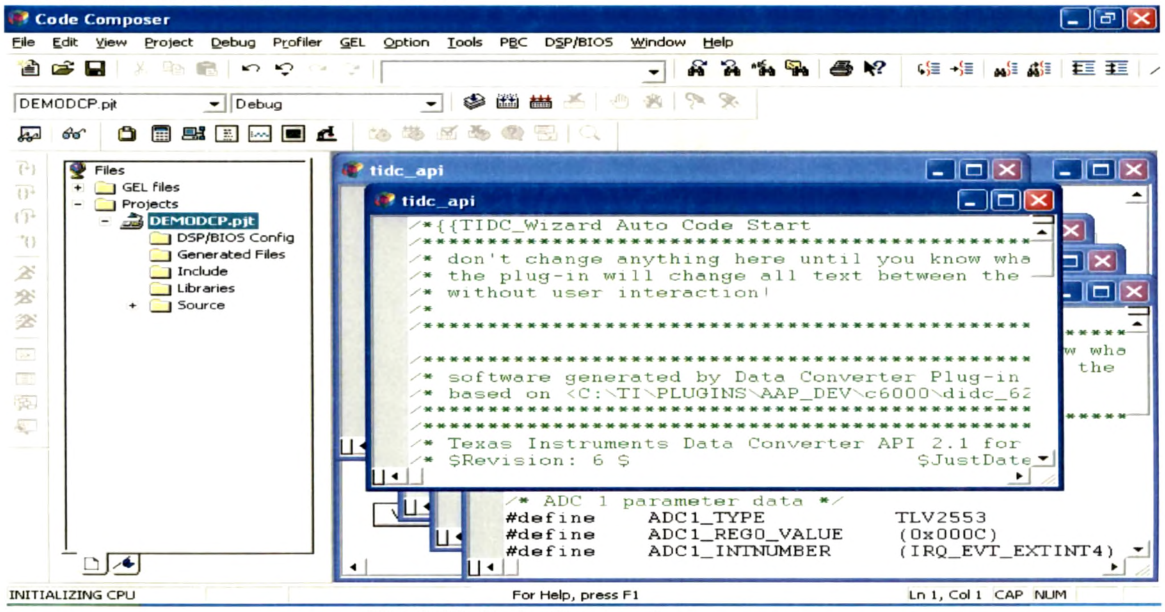
Use the WRITE FILES button to generate C source code files now.
This button will only be active if you previously added a device to the system.

Check the 'Show created files?' option above in order to view all changed and added files.

Write Files

Fig 8.29 Files Page

Click on the *Write Files* button to start the file creation process. Files as in the Fig 8.30 are generated. The *driver source code file* and the *driver header file* are not generated if the a gray-colored converter in the tree is selected



```
/*{{TIDC_Wizard Auto Code Start
/*****************************************************
/* don't change anything here until you know wha
/* the plug-in will change all text between the
/* without user interaction!
/*
/*****************************************************

/*****************************************************
/* software generated by Data Converter Plug-in
/* based on <C:\TI\PLUGINS\AAP_DEV\c6000\didc_62
/*****************************************************
/*****************************************************
/* Texas Instruments Data Converter API 2.1 for
/* $Revision: 6 $                      $JustDate
```

```
/* ADC 1 parameter data */
#define      ADC1_TYPE              TLV2553
#define      ADC1_REG0_VALUE        (0x000C)
#define      ADC1_INTNUMBER         (IRQ_EVT_EXTINT4)
```

Fig 8.30 Code Generation

The current version of CCS already includes support for 11 data converters. Further devices currently under development will be integrated gradually into the plug-in. An installation program, updated with the latest plug-in, is available for downloads from TI's Web site at www.ti.com/sc/dcplug-in Program examples for some of the DSP DSKs and data converter EVMs are also available. Future standard software interfaces for ADCs and DACs are currently under development

8.3 APF Controller Software

For designing the control circuit of active power filter the MATLAB / SIMULINK is used. The design is based on the instantaneous active reactive power theory. The control circuit contains two major parts

1. Generation of 3-ø compensating currents
2. Generation of 6 PWM signals from these compensating currents

For DSP implementation, the program should be in DSP assembly language. We can generate the DSP assembly language code with the help of MATLAB / SIMULINK in two ways.

- We can convert the simulink model (.mdl) file into the C source code with the help of Real Time Workshop and then this C source code can be loaded into the Code Composer Studio (CCS) of Texas Instruments.
- The other procedure is we can use the Code Composer Studio Link (CCS Link) given in MATLAB and we can directly convert the simulink model file into DSP assembly language code.

These both processes are discussed in this chapter in detail. The DSP code can be generated with the help of any one of the above method.

8.3.1. Code Generation Using Real Time Workshop & CCS

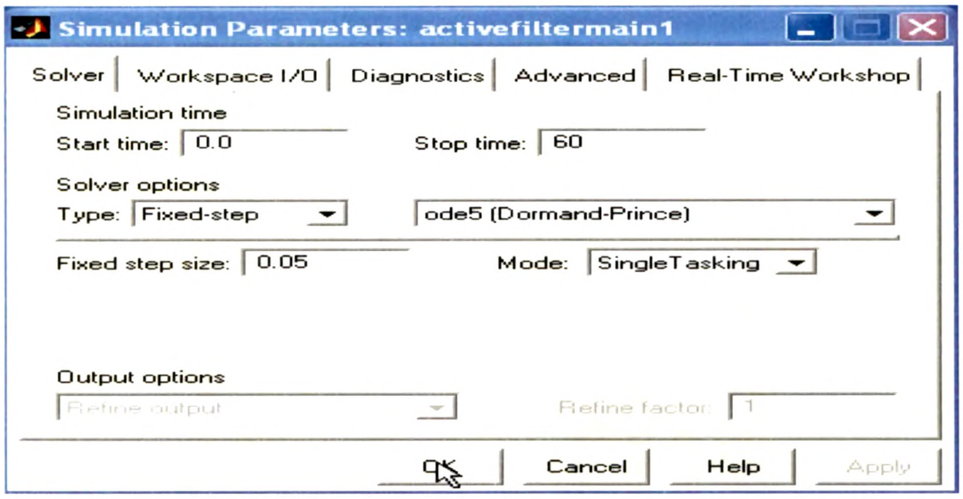
The C code can be generated from the Real Time Workshop given in the simulink RTW imposes certain requirements and restrictions on the model from which code is generated. Many of these, for example, the use of variable-step solvers, are target specific.

- **SETTING PROGRAM PARAMETERS**

To generate code correctly from your model, you must change some of the simulation parameters. In particular, note that generic real-time (GRT) and most other targets require that the model specify a fixed-step solver. To set parameters, use the **Simulation Parameters** dialog box (Fig. 8.31) as follows:

- o From the **Simulation** menu, choose **Simulation Parameters**. The **Simulation Parameters** dialog box opens.
- o Click the **Solver** tab and enter the following parameter values on the Solver pane.
- o **Start Time**: 0.0; **Stop Time**: 60;
- o **Solver options:** set **Type** Fixed-step. Select the ode5 (Dormand-Prince).
- o **Fixed step size**: 0.05.**Mode**: Single Tasking.
- o Click Apply. Then click ok to close the dialog box.

o Save the model. Simulation parameters persist with the model, for use in future sessions.



Simulation Parameters: activefiltermain1

Solver | Workspace I/O | Diagnostics | Advanced | Real-Time Workshop

Simulation time
Start time: 0.0 Stop time: 60

Solver options
Type: Fixed-step ode5 (Dormand-Prince)
Fixed step size: 0.05 Mode: SingleTasking

Output options
Refine output Refine factor: 1

Fig 8.31 SETTING PROGRAM PARAMETERS

- **SELECTING THE TARGET CONFIGURATION**

To select the GRT target:

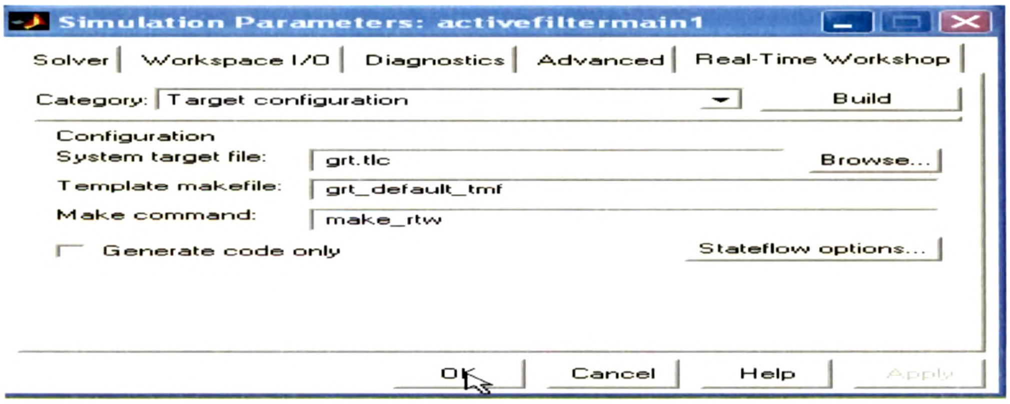
o From the **Simulation** menu, choose **Simulation Parameters**. The **Simulation Parameters** dialog box opens.

o Click on the **Real-Time Workshop** tab of the **Simulation Parameters** dialog box. The Real-Time Workshop pane activates.

o The Real-Time Workshop pane has several parts, which are selected via the Category menu.

o Click the **Browse** button next to the **System target file** field. This opens the System Target File Browser, illustrated below. The browser displays a list of all currently available target configurations.

o On target configuration selection, RTW automatically chooses the appropriate system target file, template make file and make command. (Fig. 8.32)



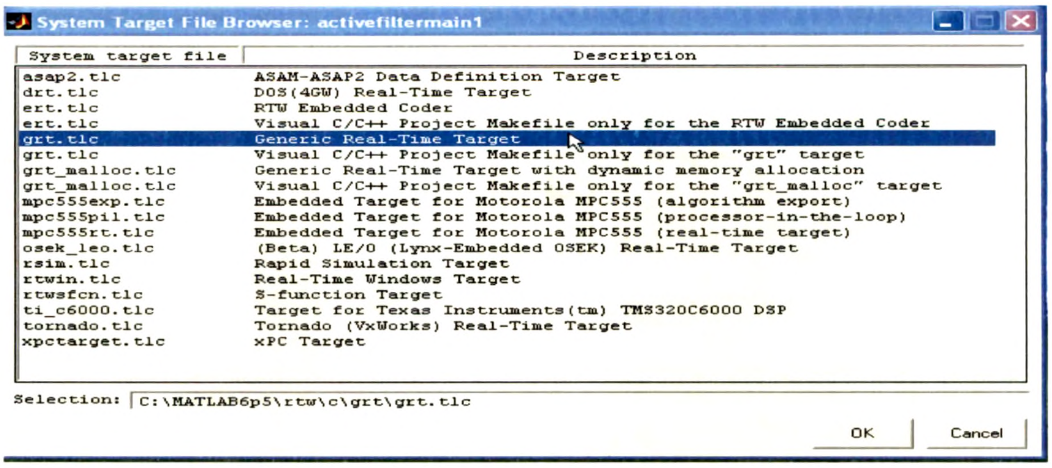
Simulation Parameters: activefiltermain1

Solver | Workspace I/O | Diagnostics | Advanced | Real-Time Workshop

Category: Target configuration

Configuration
System target file: grt.tlc
Template makefile: grt_default_tmf
Make command: make_rtw
Generate code only

Fig 8.32 REAL TIME WORKSHOP PANE


System Target File Browser: activefiltermain1

System target file	Description
asap2.tlc	ASAM-ASAP2 Data Definition Target
drt.tlc	DOS(4GW) Real-Time Target
ert.tlc	RTW Embedded Coder
ert.tlc	Visual C/C++ Project Makefile only for the RTW Embedded Coder
grt.tlc	Generic Real-Time Target
grt.tlc	Visual C/C++ Project Makefile only for the "grt" target
grt_malloc.tlc	Generic Real-Time Target with dynamic memory allocation
grt_malloc.tlc	Visual C/C++ Project Makefile only for the "grt_malloc" target
mpc555exp.tlc	Embedded Target for Motorola MPC555 (algorithm export)
mpc555pil.tlc	Embedded Target for Motorola MPC555 (processor-in-the-loop)
mpc555rt.tlc	Embedded Target for Motorola MPC555 (real-time target)
osek_leo.tlc	(Beta) LE/O (Lynx-Embedded OSEK) Real-Time Target
rsim.tlc	Rapid Simulation Target
rtwin.tlc	Real-Time Windows Target
rtwsfcn.tlc	S-function Target
ti_c6000.tlc	Target for Texas Instruments(tm) TMS320C6000 DSP
tornado.tlc	Tornado (VxWorks) Real-Time Target
xpctarget.tlc	xPC Target

Selection: C:\MATLAB6p5\rtw\c\grt\grt.tlc

Fig 8.33 SYSTEM TARGET FILE BROWSER

- From the list of available configurations, select Generic Real-Time Target (fig. 8.33) and then click OK.
- The Real-Time Workshop panes now display the correct system target file (grt.tlc), template makefile (grt_default_tmf), and make command (make_rtw).
- Save the model.

• BUILDING AND RUNNING THE PROGRAM

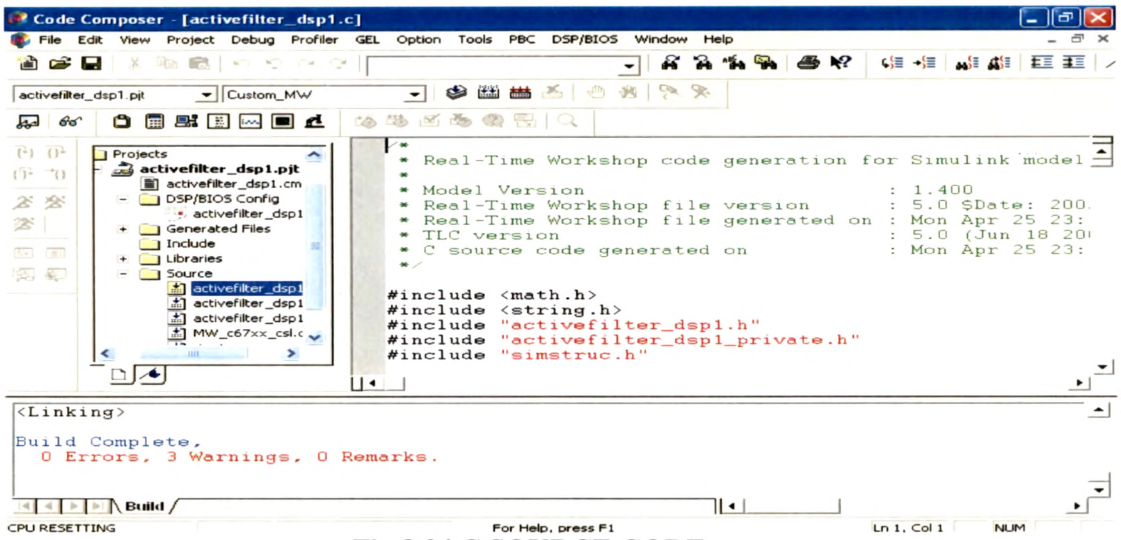
To build and run the program:

- Click the **Build** button in the **Simulation Parameters** dialog box to start the build process The final message will be
- # # # successful completion of Real-Time Workshop build procedure for model: activefiltermain1
- To observe the contents of the working directory after the build, type the **dir** command from the MATLAB command window.
  - **dir**
  - **.**
  - **.. activefiltermain1.exe activefiltermain1.mdl**
  - **activefiltermain1_grt_rtw**
- To run the executable from the MATLAB command window enter;
  >> activefiltermain1
- By typing this it will give the following message
  - **** starting the model ****
  - **** created activefiltermain1.mat ****

8.3.2. Code Generation: Using CODE COMPOSER STUDIO)

- In the code composer studio select project – new. Give some name. Say new3.pjt
- In this project select project – add files to project. A browser opens, select the required path. You only need to add 'C' programs. Header files will be added automatically in the include subfolder while checking dependencies.

- Select File – New – DSP – BIOS configuration, and select appropriate DSP configuration. For our case c6711. Save this as new3.cdb
- Go to project – add files to project and add new3.cdb and new3.cmd. new3cfg.sf6 will be automatically added.
- Go to project – compile. Remove all compilation errors.
- Go to project – build. If the program contains no syntax error and output file will be made, new3.out
- Go to File – Load program, select new3.out as the output file to be loaded in DSP.
- Once program is loaded in DSP go to Debug – Run. You will be able to see the results.
- Fig (8.34) shows a 'c' program in code composer studio.

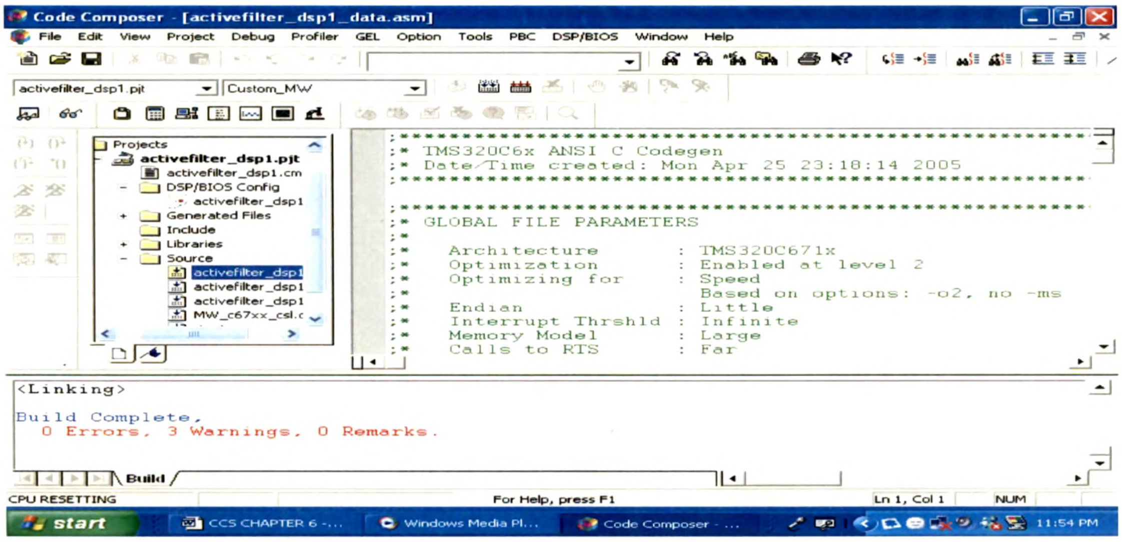


```
/*
 * Real-Time Workshop code generation for Simulink model
 *
 * Model Version                          : 1.400
 * Real-Time Workshop file version        : 5.0 $Date: 200
 * Real-Time Workshop file generated on   : Mon Apr 25 23:
 * TLC version                            : 5.0 (Jun 18 20
 * C source code generated on             : Mon Apr 25 23:
 */

#include <math.h>
#include <string.h>
#include "activefilter_dsp1.h"
#include "activefilter_dsp1_private.h"
#include "simstruc.h"
```

```
<Linking>

Build Complete,
  0 Errors, 3 Warnings, 0 Remarks.
```

Fig 8.34 C SOURCE CODE

- Fig (8.35) shows its assembly language version which is created by CCS.



```
;******************************************************************
;* TMS320C6x ANSI C Codegen
;* Date/Time created: Mon Apr 25 23:18:14 2005
;******************************************************************

;******************************************************************
;* GLOBAL FILE PARAMETERS
;*
;*   Architecture          : TMS320C671x
;*   Optimization          : Enabled at level 2
;*   Optimizing for        : Speed
;*                           Based on options: -o2, no -ms
;*   Endian                : Little
;*   Interrupt Thrshld     : Infinite
;*   Memory Model          : Large
;*   Calls to RTS          : Far
```

```
<Linking>

Build Complete,
  0 Errors, 3 Warnings, 0 Remarks.
```

Fig 8.35 ASSEMBLY LANGUAGE CODE

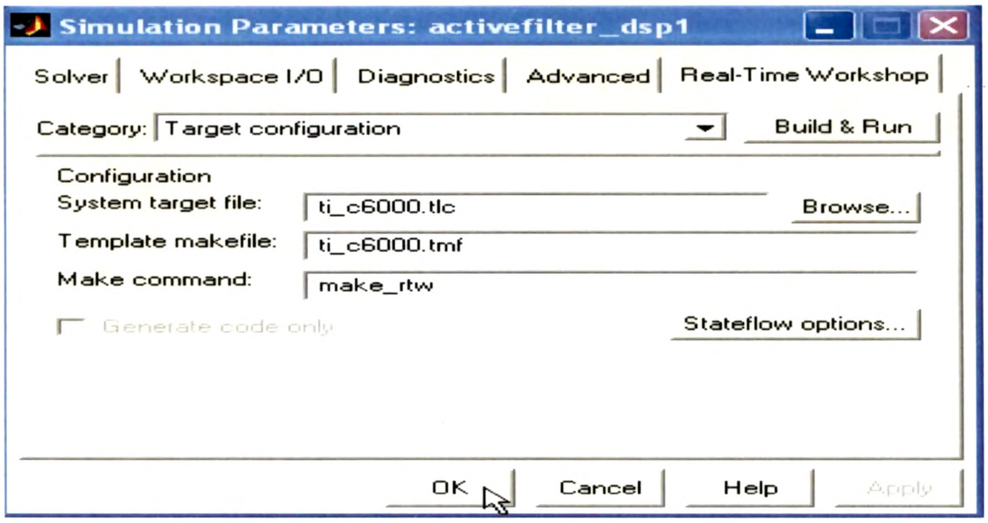
8.3.3 Code Generation Using CCS Link

For creating assembly language program using CCS link the rules are as follows:

- The solver to be used should be fixed step.
- The simulink model file should be discrete.

The steps for converting simulink model file into DSP assembly language code using CCS link are as follows:

- From the **Simulation** menu, choose **Simulation Parameters**. The **Simulation Parameters** dialog box (Fig 8.36)opens.



Category: Target configuration

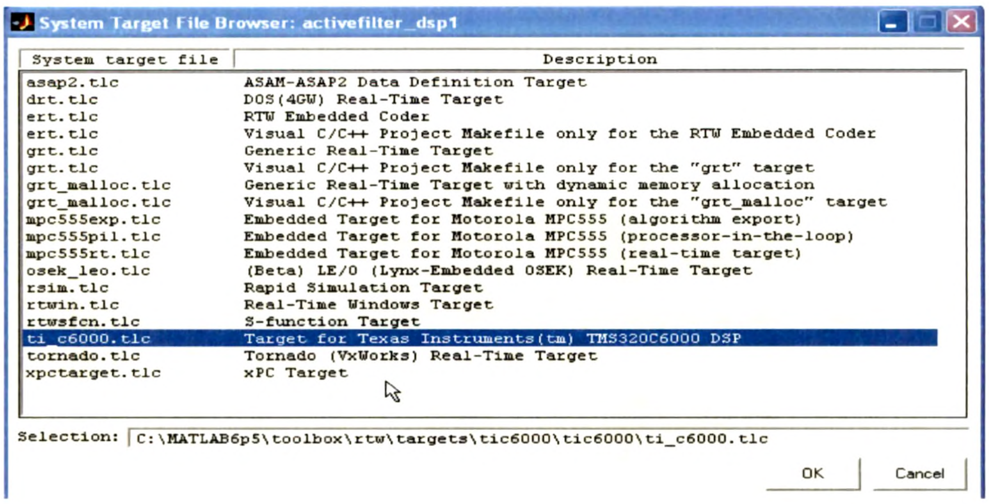
Configuration
System target file: ti_c6000.tlc
Template makefile: ti_c6000.tmf
Make command: make_rtw
☐ Generate code only

Fig 8.36 Simulation Parameters: Dialog Box

- Click the **Solver** tab and enter the following parameter values on the Solver pane.
  Start Time: 0.0; **Stop Time**: 60;
  Solver options: set **Type** Fixed-step. Select the ode5 (Dormand - Prince).
  Fixed step size: 0.05.**Mode**: Single Tasking.

- Click Apply. Then click ok to close the dialog box.
- Save the model. Simulation parameters persist with the model, for use in future sessions.

- Click on the **Real-Time Workshop** tab of the **Simulation Parameters** dialog box. The Real-Time Workshop pane activates. 3
- The Real-Time Workshop pane has several parts, which are selected via the Category menu.
- Click the **Browse** button next to the **System target file** field. This opens the System Target File Browser (Fig 8.37).



System Target File Browser: activefilter_dsp1

System target file	Description
asap2.tlc	ASAM-ASAP2 Data Definition Target
drt.tlc	DOS(4GW) Real-Time Target
ert.tlc	RTW Embedded Coder
ert.tlc	Visual C/C++ Project Makefile only for the RTW Embedded Coder
grt.tlc	Generic Real-Time Target
grt.tlc	Visual C/C++ Project Makefile only for the "grt" target
grt_malloc.tlc	Generic Real-Time Target with dynamic memory allocation
grt_malloc.tlc	Visual C/C++ Project Makefile only for the "grt_malloc" target
mpc555exp.tlc	Embedded Target for Motorola MPC555 (algorithm export)
mpc555pil.tlc	Embedded Target for Motorola MPC555 (processor-in-the-loop)
mpc555rt.tlc	Embedded Target for Motorola MPC555 (real-time target)
osek_leo.tlc	(Beta) LE/O (Lynx-Embedded OSEK) Real-Time Target
rsim.tlc	Rapid Simulation Target
rtwin.tlc	Real-Time Windows Target
rtwsfcn.tlc	S-function Target
ti_c6000.tlc	Target for Texas Instruments(tm) TMS320C6000 DSP
tornado.tlc	Tornado (VxWorks) Real-Time Target
xpctarget.tlc	xPC Target

Selection: C:\MATLAB6p5\toolbox\rtw\targets\tic6000\tic6000\ti_c6000.tlc

Fig. 8.37 Target File Browser

On selecting target, RTW automatically chooses the appropriate system target file, template make file, and make command. From the list of available configurations, select Target for Texas Instruments (tm) TMS320C6000 DSP and then click OK.

- The Real-Time Workshop panes now display the correct system target file (ti_c6000.tlc), template makefile (ti_c6000.tmf), and make command (`make_rtw`).

- Click the **Build & Run** button in the **Simulation Parameters** dialog box to start the build process. The final message will be

### Generating the DSP/BIOS configuration file...

### Creating project in Code Composer Studio(tm)

### Activefilter_dsp1.mk which is generated from

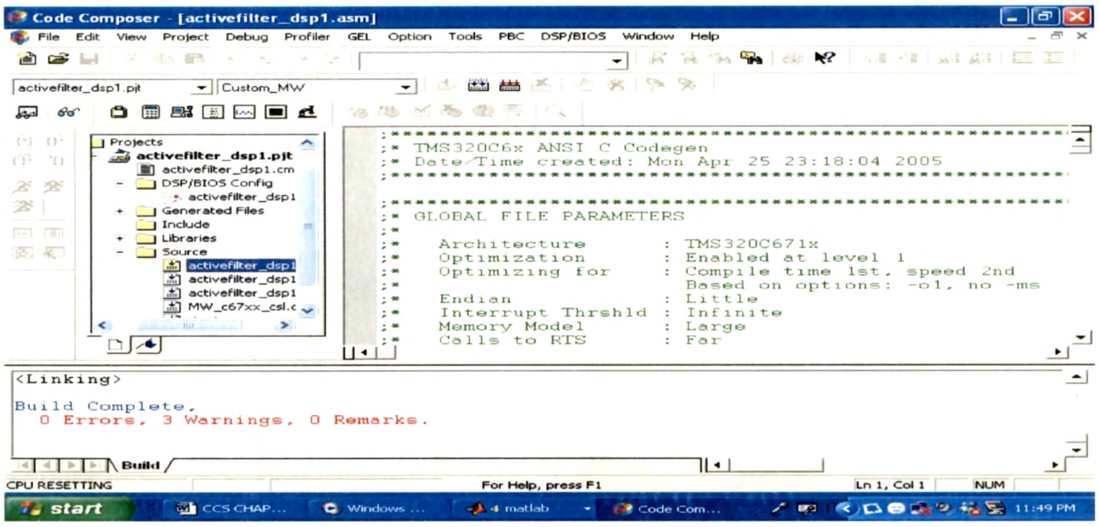
C:\MATLAB6p5\toolbox\rtw\targets\tic6000\tic6000\ti_c6000.tmf is up to date

### Building Code Composer Studio(tm) project...

### Build complete

- It directly opens the code composer studio.
- It generates the .out file, .asm files and gathers all the information.
- Go to File – Load program, select activefilter_dsp1.out as the output file to be loaded in DSP.
- Once program is loaded in DSP go to Debug – Run. You will be able to see the results.

- Fig. 8.38 shows the CCS window showing the activefilter_dsp1.pjt



```
;******************************************************************
;* TMS320C6x ANSI C Codegen
;* Date/Time created: Mon Apr 25 23:18:04 2005
;******************************************************************

;******************************************************************
;* GLOBAL FILE PARAMETERS
;*
;*   Architecture       : TMS320C671x
;*   Optimization       : Enabled at level 1
;*   Optimizing for     : Compile time 1st, speed 2nd
;*                        Based on options: -o1, no -ms
;*   Endian             : Little
;*   Interrupt Threshld : Infinite
;*   Memory Model       : Large
;*   Calls to RTS       : Far
```

```
<Linking>

Build Complete,
  0 Errors, 3 Warnings, 0 Remarks.
```

Fig 8.38 CCS WINDOW: activefilter_dsp1.pjt

8.4 I/O Port Design

The DSK provides an asynchronous expansion memory interface connector (J1) to add memory or memory-mapped devices via a daughterboard. The expansion memory interface is mapped into the lower 3M bytes of the DSP's 4M-byte asynchronous CE1 memory space. Expansion memory in the CE1 space is addressed from 0x1000000–12FFFFF in MAP 0 and 0x1400000–16FFFFF in MAP 1 mode. The upper 1M bytes of the CE1 memory space is allocated for onboard peripherals. This division of the CE1 memory space allows both the onboard devices and the expansion memory interface to coexist without conflicts.

The CPLD provides transceiver control logic that prevents the expansion memory space from conflicting with the onboard use of the CE1 space. The CPLD monitors the CE1 signal along with the upper address signals (EA [21:20]) to determine when the lower 3M-byte expansion memory space is being accessed and enables the expansion memory transceivers accordingly. CE1 decoding in the upper 1M byte is handled by the CPLD for control of onboard peripherals. The EMIF CE2 and CE3 memory space enables are available on the expansion peripheral interface connector (J3). These two memory spaces can also be used for asynchronous memory on the daughterboard when their respective SDRAM enable bits are not asserted in the CPLD register. The SDRAM enable bits control the SDRAM clock enables, as well as enabling the expansion memory transceivers to be turned on during CE2 and CE3 memory space accesses. This capability supports applications that do not require one or both banks of SDRAM, but need to interface to faster or additional asynchronous memory on a daughterboard.

All expansion memory interface signals are buffered using TI 'LVTH buffers/ transceivers to allow both 3.3- and 5-V devices to be used on the daughterboard and to isolate the daughterboard from the onboard EMIF. The three memory space enables (CE1–CE3) are buffered versions of the DSP outputs and are not generated by decode logic. This allows fast daughterboard logic to be used as required for the application without incurring additional delay. The expansion memory transceivers isolate the daughterboard and onboard data busses to prevent bus contention.

The 'C6711 DSK provides two expansion connectors that allow a daughterboard to be connected to the board. Daughter boards can be used to extend the capabilities of the DSK and to provide custom and application-specific I/O. The 'C6711 DSK's expansion memory and peripheral interfaces are provided with two dual-row, 80-pin connectors. These surface-mount connectors are low profile and have a 0.050-inch (1.27-mm) pitch. The recommended mating connectors provide 0.465-inch board spacing, allowing ample space for daughterboard components. The expansion memory interface provides the DSP's asynchronous EMIF signals to a daughterboard. External asynchronous memories and memory mapped devices can be added to the DSK, including nonvolatile memory that can be used to boot the DSK upon reset.

8.5. HARDWARE SETUP

In this project work the control signal generator unit for active power filter is implemented using digital signal processor. The schematic diagram for active power filter is shown in fig. 8.39.

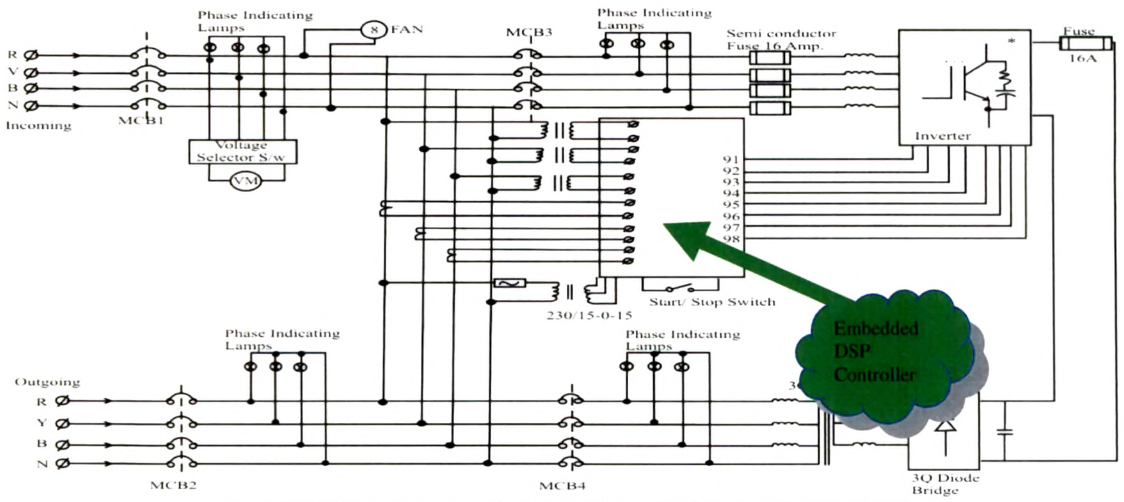


Fig 8.39 SCHEMATIC OF ACTIVE POWER FILTER

The active filter controller shown in fig. 8.39 is implemented using DSP in this project work. The steps to generate DSP code are shown in previous sections. For implementation of the DSP controller the hardware tools used are TMS320C6711 DSP Starter Kit, ADS 8364 EVM (6 input ADC); TLV 2553 EVM (11 input ADC); TLV 5614 EVM (4 output DAC) and 5-6 K Interface card. In order to study effect of resolution of ADC the DSP controller for this project was setup in two configurations: Low resolution setup (using 11 input ADC) and High resolution setup (using 6 input ADC)

8.5.1 High resolution setup (using 11input ADC)

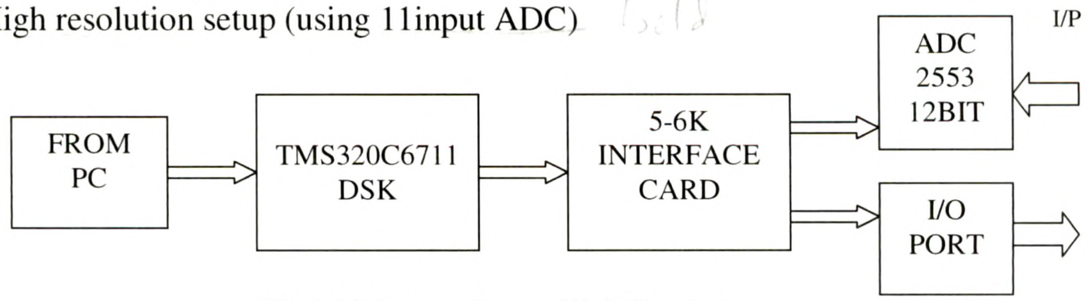


Fig 8.40 System Setup: High Resolution

Fig. 8.40 depicts the high resolution setup. TLV2553 which is a 12 bit serial ADC is used. Since the ADC is 12 bit, the resolution it offers is less. This ADC can take 11 inputs so there is no need of hysterisis card in this setup. The reference currents for the generation of firing pulses can be given directly to the ADC as analog inputs in this setup. The firing signal can be given to IGBT driver circuit using I/O port in this setup. The I/O port design is shown in previous section.

8.5.2 Low resolution setup (using 6 input ADC)

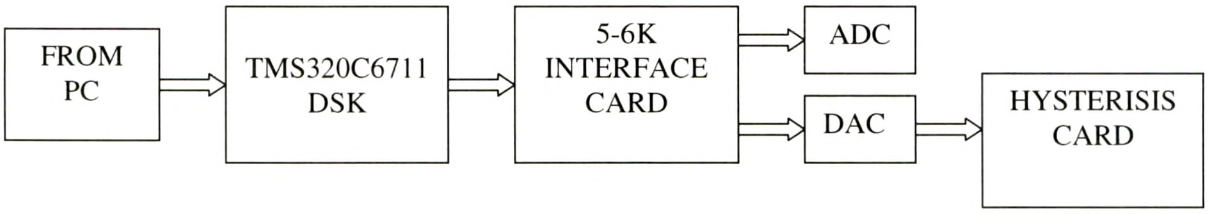


_Fig 8.41 System Setup: Low Resolution

In fig. 8.41 the low resolution setup is shown. In this setup the ADC used is ADS 8364 which is a 16 bit parallel ADC with 6 channels. Here the reference currents can not be given as analog inputs so they are taken directly from the system and the DAC is used to convert them in to analog values. The DAC used is TLV5614. The firing pulses are generated using analog hysterisis card in this setup. The reference currents are given through software in this setup.

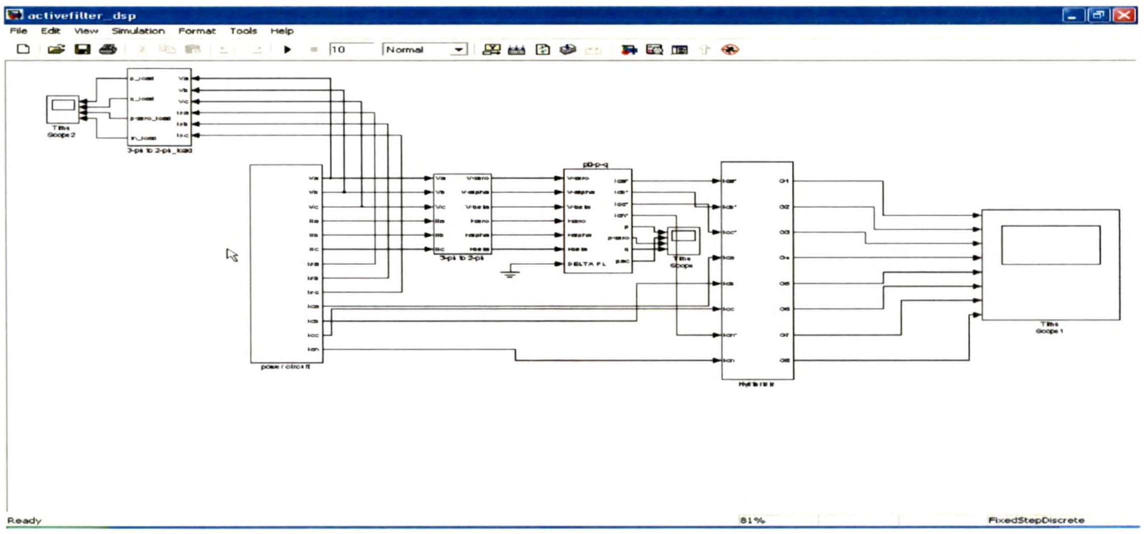


Fig 8.42 SIMULINK model

8.6 APF Controller Software

To implement the DSP controller the interfacing of ADC and DSP should be done with the help of some interface system. The flow chart (Fig 8.43) depicts the working of system and implementation steps.

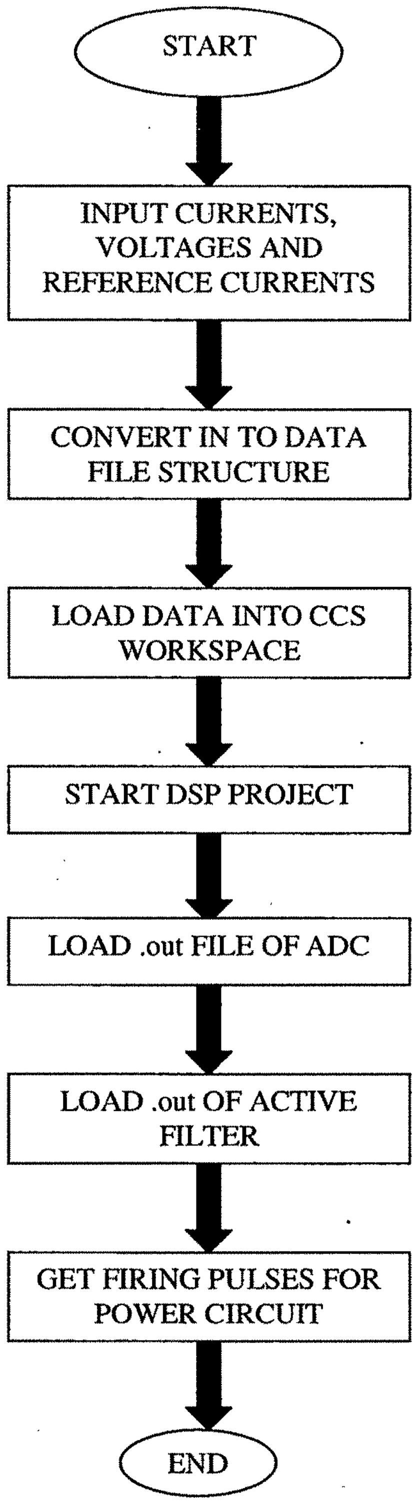

START
INPUT CURRENTS, VOLTAGES AND REFERENCE CURRENTS
CONVERT IN TO DATA FILE STRUCTURE
LOAD DATA INTO CCS WORKSPACE
START DSP PROJECT
LOAD .out FILE OF ADC
LOAD .out OF ACTIVE FILTER
GET FIRING PULSES FOR POWER CIRCUIT
END

Fig 8.43 System Flowchart: APF Controller

Fig 8.43 depicts the system model. The firing pulses which are then to be given to the power circuit of active power filter. Fig 8.44 and 8.45 shows the waveforms for firing circuits with and with out loading.

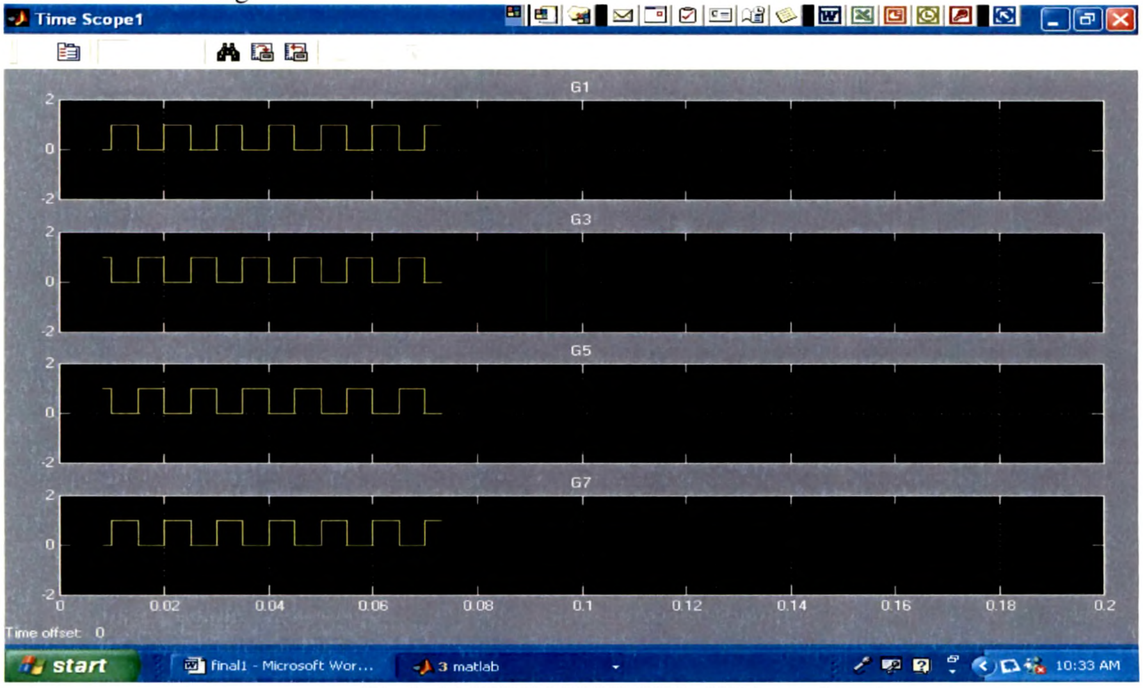


Fig 8.44 FIRINGPULSES: Without Load

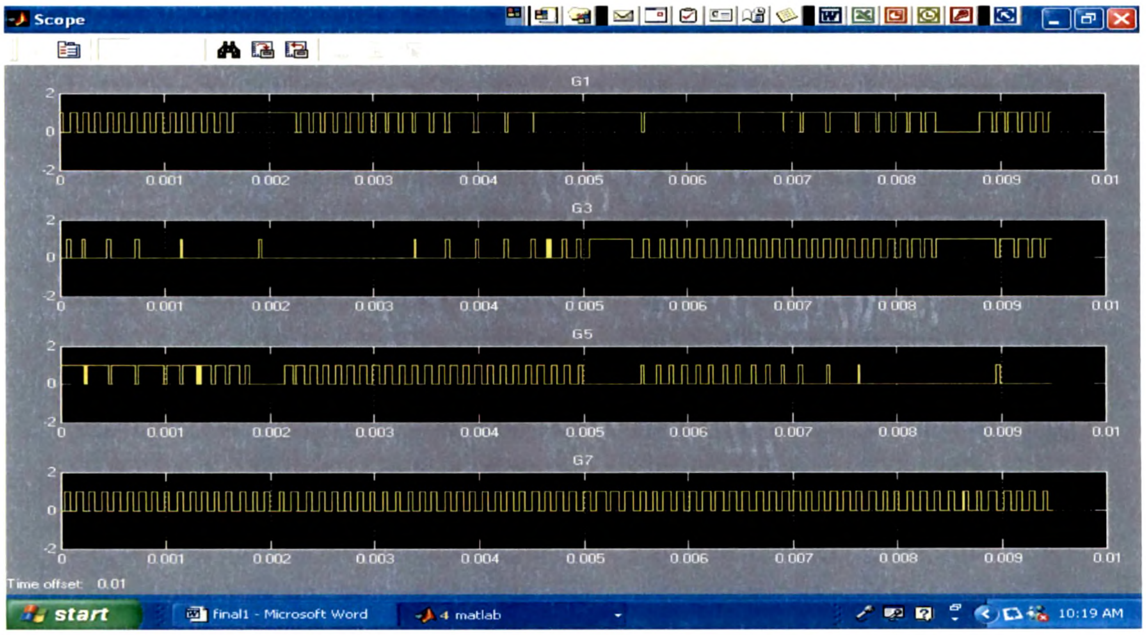


Fig 8.45 FIRING PULSES: with load (for compensation)